Power Management Strategies for Wired Communication Networks

by

Qun Yu

Bachelor of Engineering

Beijing Information Science and Technology University

2000

Submitted to the Graduate Faculty of

the School of Computing and Information in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Qun Yu

It was defended on

December 13th, 2019

and approved by

Dr. Taieb Znati, School of Computing and Information, University of Pittsburgh

Dr. Martin B.H. Weiss, School of Computing and Information, University of Pittsburgh

Dr. Prashant Krishnamurthy, School of Computing and Information, University of Pittsburgh

Dr. Daniel Mosse, School of Computing and Information, University of Pittsburgh

Dr. Balaji Palanisamy, School of Computing and Information, University of Pittsburgh

Dissertation Director:

Dr. Taieb Znati, School of Computing and Information, University of Pittsburgh

Power Management Strategies for Wired Communication Networks

Qun Yu, PhD

University of Pittsburgh, 2020

With the exponential traffic growth and the rapid expansion of communication infrastructures worldwide, energy expenditure of the Internet has become a major concern in IT-reliant society. This energy problem has motivated the urgent demands of new strategies to reduce the consumption of telecommunication networks, with a particular focus on IP networks. In addition to the development of a new generation of energy-efficient network equipment, a significant body of research has concentrated on incorporating power/energy-awareness into network control and management, which aims at reducing the network power/energy consumption by either dynamically scaling speeds of each active network component to make it capable of adapting to its current load or putting to sleep the lightly loaded network elements and reconfiguring the network. However, the fundamental challenge of greening the Internet is to achieve a balance between the power/energy saving and the demands of quality-of-service (QoS) performance, which is an issue that has received less attention but is becoming a major problem in future green network designs. In this dissertation, we study how energy consumption can be reduced through different power/energy- and QoS-aware strategies for wired communication networks.

To sufficiently reduce energy consumption while meeting the desire QoS requirements, we introduce several different schemes combing power management techniques with different scheduling strategies, which can be classified into experimental power management (EPM) and algorithmic power management (APM). In these proposed schemes, the power management techniques that we focus on are speed scaling and sleep mode. When the network processor is active, its speed and supply voltage can be decreased to reduce the energy consumption (speed scaling), while when the processor is idle, it can be put in a low power mode to save the energy consumption (sleep mode). The resulting problem is to determine how and when to adjust speeds for the processors, and/or to put a device into sleep mode. In this dissertation, we first discuss three families of dynamic voltage/frequency

scaling (DVFS) based, QoS-aware EPM schemes, which aim to reduce the energy consumption in network equipment by using different packet scheduling strategies, while adhering to QoS requirements of supported applications. Then, we explore the problem of energy minimization under QoS constraints through a mathematical programming model, which is a DVFS-based, delay-aware APM scheme combing the speed scaling technique with the existing rate monotonic scheduling policy. Among these speed scaling based schemes, up to 26.76% dynamic power saving of the total power consumption can be achieved. In addition to speed scaling approaches, we further propose a sleep-based, traffic-aware EPM scheme, which is used to reduce power consumption by greening routing light load and putting the related network equipment into sleep mode according to twelve flow traffic density changes in 24-hour of an arbitrarily selected day. Meanwhile, a speed scaling technique without violating network QoS performance is also considered in this scheme when the traffic is rerouted. Applying this sleep-based strategy can lead to power savings of up to 62.58% of the total power consumption.

Table of Contents

List of Tables

List of Figures

Preface

This dissertation is the result of interdisciplinary research in network telecommunication and computer science, which is a research topic I embraced since becoming a graduate student. During this long journey, I have a special appreciation for my academic advisor, Dr. Taieb Znati, for his long-time support and encouragement in my academic studies. I admire his patience in modifying my drafts and manuscripts as well as his rigorous academic attitude. Without his support, I could not have completed this dissertation.

I thank the committee members who provided me a vast amount of suggestions to improve my dissertation. I also appreciate Dr. Wang Yang, who insisted that I focus on the writing style of my proof. By following his suggestions, I spotted the logic flaws in my proof, which significantly elevated the quality of the math parts.

Last, I welcome the support from my lab mates and residential community. I am very grateful to Dr. Xiao Ma from TELE at Pitt, who enriched my research and study methods. A special thank you goes to Dr. Yang Song from ECON at Pitt for her inspiration and encouragement. Finally, thanks go to my family: my parents, my husband, and my younger sister.

1.0 INTRODUCTION

In the past few years, Information and Communication Society (ICS) has experienced unprecedented growth in the amount of information being processed, stored, and transferred over the Internet [Pierson, 2015]. Due to the constantly increasing number of customers and new services being offered, data traffic volume doubles every eighteen months according to Moore's law [Zhang et al., 2008b], which causes an even larger increase in number and capacity of network equipment to guarantee the QoS requirements of supported applications. Recent advances in semiconductor technology, which enable higher parallelism and increase clock frequencies, paved the way to a new generation of powerful routers. These advances, however, come at a costly price of increased power consumption [Chabarek et al., 2008]. According to figures in 2007, Information and Communications Technology (ICT) power requirement was estimated to be within a range from 2% to 10% of global power consumption [Lubritto et al., 2008; Koomey et al., 2007], while the energy demand of network equipment, excluding servers in data centers, was around 22 GW with expectations of reaching 95 GW in 2020 [Vereecken et al., 2008]. Other data related to energy consumption show that the telecom operators' demand grew from 150 TWh/y in 2007 to 260 TWh/y in 2012, around 3% of the total worldwide need [Lambert et al., 2012]. Finally, other work focusing on a single Internet Service Provider (ISP) shows that the energy consumed by the largest ISPs, such as AT&T or China Mobile, reached 11 TWh per year in 2010, while medium-sized ones like Telecom Italia and GRNET were expected to approach 400 GWh in 2015 [Bolla et al., 2012].

With the dramatic increase in energy expenditures of the Internet, the power consumption of routers is becoming a bottleneck with the growing traffic, despite all of the semiconductor technology's upgrades [Lange et al., 2009; Tucker et al., 2009]. In 2009, the backbone energy consumption accounted for less than 10% of the overall network energy consumption, but this percentage was expected to increase to 40% in 2017 [Lange et al., 2009], and reach up to or even exceed 50% in 2020; thus, it will become unsustainable [Group, 2007; Lange et al., 2011; Hinton et al., 2011]. In addition, another problem is that the energy consumption of

current IP networks is not proportional to the utilization level. Traditionally, networking systems are designed and dimensioned according to principles that are inherently in opposition with green networking objectives, namely over-provisioning and redundancy [Bianzino et al., 2012]. On the one hand, due to the lack of QoS support from the Internet architecture, over-provisioning is a common practice: networks are dimensioned to sustain peak hour traffic, with extra capacity to allow for unexpected events. As a result, during low traffic periods, over-provisioned networks are also over-energy-consuming. Moreover, for resiliency and fault-tolerance, networks are also designed in a redundant manner. Devices are added to the infrastructure with the sole purpose of taking over the duty when another device fails, which further adds to the overall energy consumption. Therefore, even in low or no usage context, network equipment consumes energy at a high level [Pierson, 2015].

For these reasons, the key to green network designs is to seek effective strategies that incorporate power/energy-awareness into network control and management to reduce power/energy consumption, without compromising either the quality of service or the network reliability [Bolla et al., 2011b; Bianzino et al., 2012; Zeadally et al., 2012]. Currently, two of the most exciting dynamic power management (DPM) techniques are **speed scaling** and **sleep mode** [Recupero et al., 2013]. The former is used to reduce energy consumption by dynamically adjusting clock frequencies of the active component to make it able to adapt to the current load. The latter is used to save energy by putting network elements into sleep state when they are idle or low-demand. Excessive reduction in execution rates or extended sleep periods to save energy, however, could result in severe network degradation which may lead to violation of QoS requirements of the underlying applications. Consequently, seeking effective power/energy-aware strategies by using these techniques to reduce power/energy consumption, while guaranteeing QoS performance requirements, becomes a significant challenge.

1.1 Problem Statement

The problem studied in this dissertation is sufficient energy saving or energy minimization under QoS requirements for wired communication networks, focusing on the exploration

of energy-aware strategies to sufficiently reduce and even minimize network power/energy consumption by pursuing three aspects: (i) the definitions of methodologies for power- or energy-aware networking design; (ii) the designs of power/energy management strategies to adapt the network energy consumption to current traffic load; and (iii) a fine balance achievement between saving energy and adhering to QoS performance, which is an issue that has not been sufficiently studied but is becoming a major concern in future networks.

As introduced above, two DPM techniques, namely speed scaling and speed mode, which have been the conventionally effective methods to reduce energy consumption, can be used to solve this problem [Bolla et al., 2011b]. The speed (operations per second) of many devices can be decreased to lower the power consumption. This technique is called speed scaling, which usually results in a decreased energy consumption, despite the fact that the power is consumed for a longer time [1]. A popular speed scaling technique that is used in modern microprocessors is dynamic voltage/frequency scaling (DVFS). DVFS is applied to decrease the clock frequency (and with it, the voltage) that leads to reduced speed and power consumption. Speed scaling is also used in other devices, such as flash storage, hard drives, and network cards [Lee and Kim, 2010; Rao and Vrudhula, 2005]. Currently, various DVFS techniques [Mandviwalla and Tzeng, 2006; Valentini et al., 2013; Gerards, 2014; Nedevschi et al., 2008] are utilized to exploit the variations in the actual workload for dynamically adjusting the voltage and frequency of processors to achieve energy saving. Besides speed scaling, many devices support switching to various low-power sleep states to reduce energy consumption when they are idle or low-demand. This technique is called speed mode. Typically, the main challenge for seeking effective energy saving though DVFS-based techniques is to preserve the feasibility of scheduling strategies and provide performance guarantees, while reducing the energy demand of network-enabled devices through sleep mode techniques is not only about getting idle devices to sleep but also about making sure that they can respond to valid network requests, i.e. that they wake up quickly when needed. The inconvenience with sleep mode techniques are that once in a power-efficient state, bringing a component back to the active or running state requires additional energy and/or delay to serve incoming traffic load, and that constantly switching network devices or their components off and on might lead to

[1] Energy is power multiplied by time.

more energy consumption than keeping them on all the time. Consequently, how these two prominent DPM techniques can be effectively used to reduce network energy consumption, while adhering to QoS requirements, becomes the exploration goal of this research.

The following research questions are explored and studied in this dissertation:

- DVFS-based power management and QoS-aware scheduling strategies
  - What characteristics does a DVFS-based, QoS-aware scheduler have?
  - How should the aggressivity for speed scaling to save more energy be defined?
  - How can a fine balance between energy saving and network performance be achieved?
- DVFS-based power management and delay-aware optimal energy strategies
  - What characteristics does a DVFS-based, delay-aware energy minimizing schedule have?
  - How can an existing scheduling algorithm minimize network energy consumption?
  - What are the optimal speeds for speed scaling?
- Sleep-based power management and and a traffic-aware strategy
  - What characteristics does a sleep-based, traffic-aware power controller have?
  - How is a sleep-based power management strategy designed?
  - How can a speed scaling technique be combined with a sleep mode strategy?
- How can power/energy-efficient metrics be defined?

1.2 Research Overview

With the exponential traffic growth and the rapid expansion of communication infrastructures worldwide, energy expenditure of the Internet has become a major concern in IT-reliant society and how to reduce or minimize power/energy consumption has become a critical objective in the design of future networks [Lange et al., 2009]. A significant body of schemes has focused on incorporating energy-awareness into network control and management. In general, the involved power management techniques can be summarized into two categories: (i) DVFS-based techniques can reduce energy consumption by tuning the

packet processing engine frequency or voltage to adapt to the current traffic load at different levels [Mandviwalla and Tzeng, 2006; Nedevschi et al., 2008; Bolla et al., 2011c; Zhang et al., 2008a; J., 2014; Chiaraviglio et al., 2009a; Alonso et al., 2004; Cisco, 2007; Galan-Jimenez and Gazo-Cervero, 2013; Vasić and Kostić, 2010]. This, however, is at the expense of lower performance; and (ii) sleep-based techniques can achieve energy saving through turning off the network devices or components when they are idle or low-demand [Nedevschi et al., 2008; Gupta and Singh, 2003; Fisher et al., 2010; Chiaraviglio et al., 2008; Gupta and Singh, 2007a; Nordman and Christensen, 2010; J., 2014; Chiaraviglio et al., 2009a; Bianzino et al., 2010; Idzikowski et al., 2010; Zhang et al., 2010b; Chiaraviglio et al., 2009b; Sabhanatarajan et al., 2008]. It, however, could come the expense of extra more energy and longer time for sleepings and wake-ups due to frequent switching on/off. These techniques and approaches mostly aim at dynamically managing network resources, in response to traffic load, in order to minimize energy consumption and reduce congestion. However, with the advent of Internet-based multimedia communication services, such as voice over IP (VoIP), video streaming, video conferencing, etc., QoS and Quality of Experience (QoE) become more and more important. The slowdown or shutdown of a process to save energy by these power management techniques may lead to QoS violations. Consequently, the need to support the QoS requirements of these emerging applications further compounds the sufficient energy saving or energy minimization problem, calling for new power/energy- and QoS-aware strategies to traffic management and congestion control.

To address the above challenges and answer the aforementioned questions in Section 1.1, this dissertation explores several effective power/energy- and QoS-aware strategies based on two prominent DMP techniques to achieve sufficient network energy saving or minimizing network energy consumption while supporting QoS requirements.

1.2.1 DVFS-based Power Management and QoS-aware Scheduling Strategies

The power management technique that we first focus on is speed scaling. Using DVFS techniques, routers can adaptively adjust the operational frequencies of their network processor unites (NPUs), according to current conditions of the network. Excessive reduction

in execution speeds to save energy, however, could result in QoS violation of the supported applications. To address the energy-QoS dichotomy, we discuss three families of QoS-aware packet scheduling strategies in Chapter 3 to dynamically control the execution rates of network components to reduce the energy consumption in routers. The main objective of these experimental power management (EPM) schemes with different packet scheduling strategies is to sufficiently save the dynamic energy consumed by routers through rate adaption, while achieving a fine balance between significant energy saving improvements and network performance requirements. Two metrics, namely **queue length** and **link utilization**, are considered to achieve this goal [Yu et al., 2015a,b].

1.2.2 DVFS-based Power Management and Delay-aware, Optimal Energy Strategies

In addition to the EPM schemes through different DVFS-based, QoS-aware packet scheduling strategies to reduce the predictable energy consumption, the energy saving issue can also be formulated by mathematical programming models by using algorithmic power management (APM) schemes. Chapter 4 addresses a key issue of how to efficiently assign per-node delays and per-node execution speeds for a given routing path, to minimize the energy consumption of network components, while satisfying the QoS delay requirements of the supported applications. To this end, under a given scheduling policy, we explore a DVFS-based, delay-aware energy optimal strategy and its two heuristics to optimize the energy consumption of network components through rate adaption, which takes into consideration the workload and the delay requirements across the routing path. To minimize energy consumption without violating network performance, we discuss the feasible per-node delays and the current potential processor execution speeds for a given flow through analyzing the processing capacity of delay-based processors along the given routing path [Yu and Znati, 2017].

1.2.3 Sleep-based Power Management and a Traffic-aware Strategy

Chapter 5 involves another dynamic power management technique: sleep mode. In a given network, when one network element is idle or low-demand, it can be put into sleep

mode by shutting down it or its partial components, thereby achieving significant energy saving. Many related studies have shown that the base system (including chassis, switch fabric, and router processor) consumes more than half of the whole power consumption of a network router. Therefore, switching off the whole router could save more energy than shutting/slowing down its components such as line cards (LCs) over the low-demand or idle periods. To this end, our work in this chapter further explores a sleep-based, traffic-aware EPM strategy, which focuses on how to achieve network sufficient energy saving by putting redundant network equipment into sleep, thereby placing network resources in more power- and energy-efficient way. A speed scaling technique is also considered to adapt to the traffic load change without damaging the QoS performance when the selected traffic is rerouted. The discussed scheme in this chapter combines the two dynamic power management techniques, namely sleep mode and speed scaling.

1.3 Claims and Contributions

In general, the research in this dissertation aims to unify the proposed energy- and QoS-aware strategies on power management for energy problems that arise with modern network architectures. A part of the theoretical work from literature does not consider important practical restrictions. On the other hand, application-oriented research projects rarely use the existing theory. Summarizing, the general contributions of this dissertation are algorithms, schemes, strategies, and concepts which are straightforward to implement and use in practice, which can be briefly concluded as the following:

- We study the dynamic power management for wired communication networks. In Chapter 3, we propose and develop three families of DVFS-based, QoS-aware packet scheduling schemes to reduce the network energy consumption through different speed scaling strategies, while adhering to the QoS requirements of the routed traffic.
- Targeting energy-performance challenges, we study different metrics, such as queue length and link utilization, to control execution rates, and try to strike a fine balance among performance, energy and accuracy in Chapter 3. A holistic simulation frame-

work, including an energy consumption model, is proposed to evaluate and compare the performance of each scheme in different networking environments and traffic models.

- To address the energy minimization problem under a given scheduling policy, we further propose a DVFS-based, delay-aware strategy through an algorithmic power management approach in Chapter 4. This strategy aims at describing a mechanism that obtains minimal energy consumption by assigning feasible per-node delays and potential per-node execution speeds across a given routing path under QoS constraints. In addition, another two heuristics are also proposed and discussed.
- Going further, we explore issues and challenges in sleep-based, energy-aware strategies for wired communication networks. In Chapter 5, we propose a feasible sleep-based, traffic-aware solution, combining two techniques of speed scaling and sleep mode, to sufficiently reduce the power consumption by putting into sleep the redundant network elements and reconfiguring the network, without network QoS violation.

1.4 Structure of this Dissertation

The rest of this dissertation is organized in the following way. Chapter 2 briefly investigates and characterizes the sources of power consumption in wired communication networks, and reviews the existing dynamic power management techniques. In Chapter 3, we discuss three families of DVFS-based, QoS-aware packet scheduling schemes to reduce energy consumption while balancing a fine tradeoff between the energy saving and QoS requirements. Given an existing scheduling policy, we build up a DVFS-based, Delay-aware energy optimization framework, and further study energy minimization under QoS constraints in Chapter 4. We explore a sleep-based, traffic-aware power management strategy and its two heuristics without network QoS violation in Chapter 5. Chapter 6 concludes the dissertation and lists future work directions.

2.0 BACKGROUND

In this chapter, we first investigate the main sources of power consumption in wired IP networks by assessing the energy characteristics of key wire-networking components. Then we introduce the existing dynamic power management techniques. Our goals are to review the state of the art on energy efficiency and to drive the research fostering activities toward energy- and QoS-aware solutions for future networks.

2.1 Characteristics of Power Consumption in Wired IP Networks

Routers and switches are widely used to connect different types of high-speed networks that make up the Internet today. From a general point of view, IP routers have a similar architecture with respect to high-end switching systems, i.e. highly modular and hierarchical architectures [Pierson, 2015]. A router or a switch chassis consists of three major subsystems, namely data plane, control plane, and environmental units [Vishwanath Member et al., 2014].

- The data plan, or called the forwarding plan, is responsible for deciding how to handle the ingress packets by looking up the routing table and then sending them to the appropriate egress interfaces (or ports). Line-cards and switch fabric cards comprise key data plane elements. The line-card is used for processing and forwarding packets using its local processing subsystem and buffer spaces for processing the packets arriving along with the ingress interfaces and awaiting transmission at the egress interfaces. The switch fabric provides sufficient bandwidth for transferring packets among different line cards. It is used to receive data from the ingress line-card interfaces and switch it to an appropriate egress one(s).
- The control plan is responsible for routing-related control functions, such as generating the network map, the way to treat packets according to the different service classifications and discard certain packets. The control plane manages the routing functions, which involve communicating with other network devices, via routing protocols such as Open

Shortest Path First (OSPF), to establishing the routing tables. Routing engine cards represent the control plane elements, which run control plane protocols to populate and update the forwarding (IP and MAC address) tables.

- The environmental units are constituted of the elements that do not play a role in handling data traffic, such as the chassis power supply, fans (or air cooling), etc.

As pointed out by Tucker et al. in [Tucker et al., 2009], network devices networking in the different network portions play a central role, since they are major contributors to the energy consumption of modern networks, and the overall energy consumption in networks arises from their operational power requirements and their density. In more detail, operational power requirements arise from all the hardware (HW) elements realizing network-specific functionalities, like the ones related to data and control planes, as well as from HW elements devoted to auxiliary functionalities, such as power supply, air cooling, etc.. In this respect, the data plane certainly handles the most energy-starving and critical component in the largest part of network device architectures, since it is generally composed of special-purpose HW elements (packet processing engines, network interfaces, etc.) that have to perform per-packet forwarding operations at very high speeds. It has been reported in [Wobker, 2012; Imaizumi and Morikawa, 2010] that LCs in the data plane consume about 70% of the total router power, and the network processor units (NPUs), where packet processing engines are used, consume more than 50% of the power consumed by one line card (LC). Focusing on high-end IP routers, Tucker et al. [Tucker et al., 2009] estimated that the power consumed by the data plane weighs for 54% of the total, vs. 11% for the control plane and 35% for power and heat management, as shown in Figure 2.1. The same authors further broke out energy consumption sources at the data-plane on a per-functionality basis. Internal packet processing engines require about 60% of the power at the data plane of a high-end router, network interfaces weigh for 13%, switching fabric for 18.5% and buffer management for 8.5%. These valuable resulting estimations provide a relevant and clear indication of how and where future research efforts need to be focused in order to build next-generation green devices [Bolla et al., 2011b].

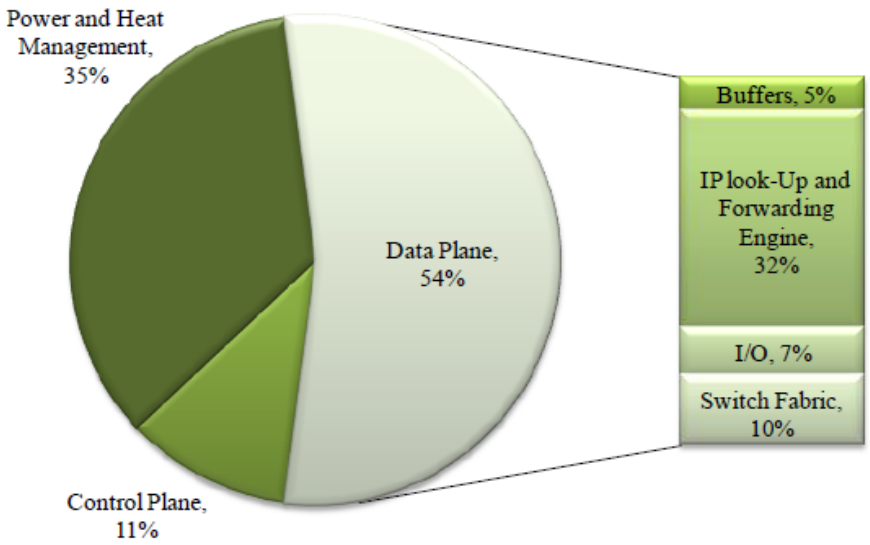

Figure 2.1: Estimation of power consumption sources in a generic platform of high-end IP router [Tucker et al., 2009].

2.2 Toward Energy- and QoS-aware Network Devices

With the rapid growth in the bandwidth of communication links for wired networks, new powerful packet switches and routers are being designed with increasing capacities and performance by exploiting recent improvements in semiconductor technologies. However, the power efficiency of the underlying technology is starting to plateau [Chabarek et al., 2008]. The slowing-down of power savings due to technology improvements will lead to an increase in power density. At the same time, the heat dissipation demands of routers are reaching the limits of traditional solutions based on air cooling. Furthermore, current router architectures are not energy-aware, in the sense that their energy consumption does not scale sensibly with the traffic load. Therefore, effective and efficient power management in networking equipment is presenting a fundamental challenge to continued bandwidth scaling on the Internet. Chabarek et al. in [Chabarek et al., 2008] analyzed two router architectures and evaluated their energy consumption under different traffic loads. The results show that the energy consumption between an idle and a heavily loaded router (with 75% of offered traffic load) vary only of 3% (about 25 W on 750 W). This happens because the router line cards,

which are the most power-consuming elements in a router, are always powered on even if they are idle. On the contrary, the energy consumption decreases to just 50% if the idle line cards are physically disconnected according to [Chabarek et al., 2008]. Such a scenario suggests that future router architectures will be energy-aware, which means that they will be able to automatically switch off or dynamically slow down subsystems (e.g. line cards, input/output ports, switching fabrics, and buffers) according to the traffic loads in order to save energy whenever possible. Therefore, energy-efficiency is the first improvement that leads to energy saving through technological innovations without affecting performance. Such solutions are usually referred to as eco-friendly solutions. While energy-awareness is the next improvement toward eco-sustainability, it refers to an intelligent technique that adapts the behavior or performance of the system to minimize its energy consumption [Pierson, 2015]. Furthermore, the fundamental problem of greening the Internet is to strike a fine balance between the demands of performance and the limitations of energy usage. Consequently, it has become a trend that the new smarter and more effective strategies enable energy awareness in the Internet architecture, taking into account the tradeoff between energy saving and network performance. Therefore, in order to realize the green Internet, an energy-oriented approach needs to define a comprehensive solution encompassing energy-efficient, energy-aware and QoS-aware aspects.

2.3 Dynamic Power Management Techniques for Wired Network Resources

In the following, the main dynamic power management techniques are detailed. Besides, the existing power/energy modeling and measuring techniques also are introduced. Using a generic power/energy model along with these power management techniques, researchers can further explore the potential impact of energy-awareness in wired communication networks.

2.3.1 Power Scaling Techniques

Power scaling techniques are aimed at achieving dynamic power management through modulating capacities of network devices resources (e.g. links bandwidths, computational

capacities of packet processing engines, etc.) or turning off low-demand (i.e.lightly loaded) or unused devices (or their subsystems) according to current traffic loads and service requirements.

2.3.1.1 Current Approaches and Concepts In general, the largest part of undertaken approaches is founded on a few base concepts, which have been generally inspired by energy saving mechanisms and dynamic power management criteria that are already partially available in computing systems. These base concepts can be categorized as Dynamic Adaptation and Smart Standby.

- **Dynamic Adaptation (or Speed Scaling)** approaches allow to modulate capacities of packet processing engines and of network interfaces according to the current traffic load and service requirements. This can be performed by using two power-aware techniques, namely, Adaptive Rate (AR) and Low Power Idle (LPI). Adaptive Rate allows a dynamic reduction of the device working rate by tuning the packet processing engine frequency or voltage. The result is a reduction of power consumption, at the price of lower performance. Low Power Idle, instead, allows reducing power consumption by putting sub-components into low power states when not sending/processing packets, and waking them up as rapidly as possible when the system needs their activity. In detail, the highest rate provides the most energy-efficient transmission while low power idle consumes minimal power. Both AR and LPI are implemented by pre-selecting a set of hardware (HW) configurations that can provide different trad-offs between packet service performance and power consumption. In a sense, AR and LPI might be jointly exploited to better fit system requirements. It is worth noting that the effectiveness of these methods strictly depends on the traffic and network characteristics. For example, the application of LPI capabilities shows better results when incoming traffic presents a high burstiness, so that the system has enough time to spend in an idle state before a new burst is received.
- **Smart Standby (or Sleep Mode)** approaches can be considered as deeper idle states: using power management primitives, devices can be turned off smartly and selectively, providing higher power saving than idle states at the price of longer wake up times [Bolla et al., 2011a]. The main issue with this approach is represented by the loss of network

connectivity. When a device is sleeping, it cannot be seen on the network, thus wake up time includes a re-connection phase, in which the device has to send signaling traffic to communicate its presence on the network and allow updating the forwarding tables. This is probably the predominant reason why the networking devices are left fully powered even when not employed. Recent solutions propose the introduction of a proxy to take charge of the host network presence during sleeping times, standby modes have to be explicitly supported with special proxying techniques able to maintain the "network presence" of sleeping nodes or components [Gunaratne et al., 2005].

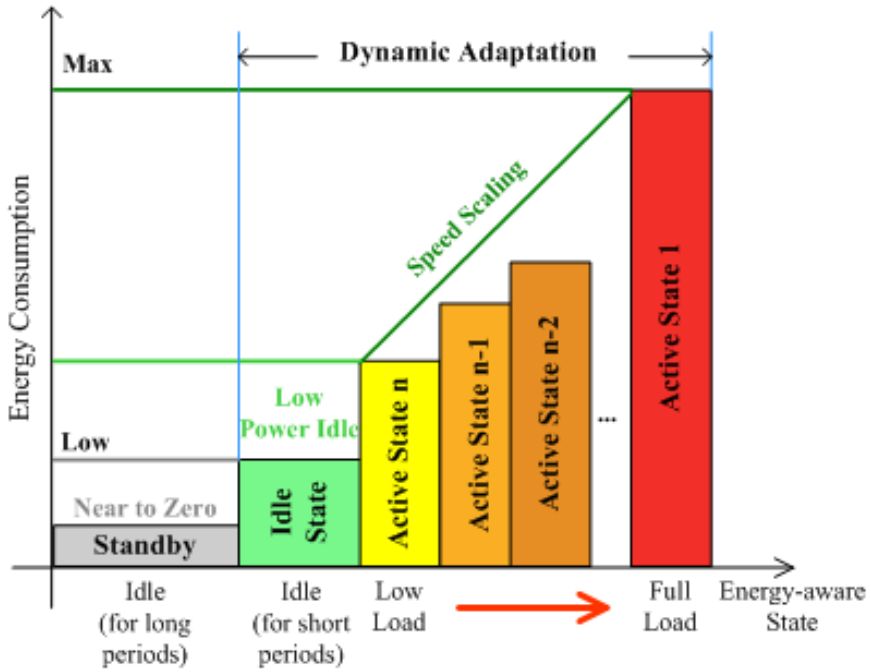

Figure 2.2: Energy-aware Profiles [Vassilakis, 2012].

Dynamic adaptation, i.e. speed scaling, approaches will enable to modulate energy absorption according to the actual workload through slowing down HW components. While approaches based on smart standby, i.e. sleep mode, will help to further cut the power consumption of unneeded or unused devices (or parts of them) by shutting down them or putting them into sleep modes. All these approaches are not mutually exclusive, and their joint adoption may eventually impact on next-generation network devices by providing "energy-aware" profiles, as depicted in Fig. 2.2. Since these approaches are founded on the idea of either

tuning device processing/transmission capacities, or of waking up the hardware upon "active" request or their combinations. Although the adoption of such green optimizations or their combinations affect network performance, the trading energy consumption for network performance is paid more and more attention by network researchers.

2.3.1.2 Dynamic Voltage/Frequency Scaling Nowadays, although the largest part of today's network equipment does not include such HW power saving capabilities, power management is a key feature in today's processors across all market segments. This is usually accomplished by scaling the clock frequency or by throttling the CPU clock, we call this technology as dynamic voltage/frequency scaling (DVFS). As one of the most promising energy-efficient technologies of speed scaling approaches, DVFS is becoming an integral part of networks for saving power. In detail, power scaling capabilities allow dynamically reducing the working rate of processing engines or of link interfaces, thereby achieving the purpose of energy saving. With DVFS technology, a CMOS-based processor can operate at different voltages/frequencies to reduce the dynamic power consumption, defined by φ^D, which can be roughly characterized as follows [Mandviwalla and Tzeng, 2006; Valentini et al., 2013]:

$$\varphi^D = a \cdot V_{DD}^2 \cdot f \cdot C \tag{2.1}$$

where a ($0 \leq a \leq 1$) is the switching activity factor, i.e. the switching activity for each clock tick, and can be considered to be the utilization of the component, V_{DD} is the supply voltage of the processor, f is the clock frequency of the processor, and C is the effective switched capacitance constant. For efficient control and smooth application of DVFS, a proportionately linear relationship is expected between the frequency and applied voltage [Gerards, 2014; Chen et al., 2012], thus, the dynamic power consumption can be modeled in terms of frequency like:

$$\varphi^D = \gamma \cdot f^3 \tag{2.2}$$

where γ is a constant parameter.

2.3.2 Power/Energy Measuring Techniques

How to measure and model the power/energy consumption is another big issue for power management of wired networks, various solutions have been proposed in the literature to evaluate at different levels the energy processors consume. Considering a set of power scalable network components, this section introduces several basic techniques used to model/measure power/energy consumption.

2.3.2.1 Power measurement When discussing power issues, two main aspects impacting power consumption need to be considered. The first, referred to as static power, arises from the bias and leakage current to support control plane, environment units, and load-independent data plane [Tucker et al., 2009; Vishwanath Member et al., 2014]. The second, referred to as dynamic power, results from the charging and discharging of the voltage saved in node capacitance of the circuit. Using Φ^S and Φ^D to denote static and dynamic power, respectively, the power Φ consumed by a router can be expressed as follows:

$$\Phi = \Phi^S + \Phi^D \tag{2.3}$$

According to [Vishwanath Member et al., 2014], the power consumption of an IP router is the sum of the power consumed by its three major subsystems, namely control plane, environmental units and data plane. Assume that $\Phi_{control}$ denotes the static power consumed by control plane, $\Phi_{environment}$ denotes the static power consumed by environment units, Φ^S_{data} denotes the static power consumed by the constant baseline components in data plane, and Φ^D_{data} denotes the dynamic power consumed by the traffic load dependent components in data plane. Accordingly, the above power components can be further expressed as:

$$\begin{aligned} \Phi^S &= \Phi_{control} + \Phi_{environment} + \Phi^S_{data} \\ \Phi^D &= \Phi^D_{data} \end{aligned} \tag{2.4}$$

2.3.2.2 A general power-aware model for router power consumption Joseph Chabarek et al. in [Chabarek et al., 2008] performed several experiments to measure the energy consumption of two different Cisco router. Both of them include their base systems

(Chassis plus router processor) and line cards, based on which it provides a generic model for router power consumption, as described in Eq. 5.5. In this model, the power consumption Φ of a router is determined by its configuration and current use. The vector X defines the chassis type of the device, the installed line cards and the configuration and traffic profile of the device. The function $\Phi^S(x_0)$ returns the power consumption of a particular chassis type, which is from control plan and environment unit, N is the number of active line cards, $\varphi^D_{data}(x_{i0})$ is the dynamic cost with a scaling factor corresponding to the traffic utilization on the router, and $\varphi^S_{data}(x_{i1})$ gives the cost of the line card in a base configuration. The cost of traffic is dependent on the configuration of the router and the amount of traffic. This model is used to formulate the optimization problem for power-aware network design.

$$\begin{aligned}\Phi(X) =& \Phi^S(x_0) + \sum_{i=0}^{N}(\varphi^D_{data}(x_{i0}, x_{i1}) + \varphi^S_{data}(x_{i1})) \\ =& \underbrace{\Phi^S(x_0) + \sum_{i=0}^{N}\varphi^S_{data}(x_{i1})}_{\Phi^S(X)} + \underbrace{\sum_{i=0}^{N}\varphi^D_{data}(x_{i0}, x_{i1})}_{\Phi^D(X)}\end{aligned} \tag{2.5}$$

2.4 Conclusions

Using the above techniques introduced in this chapter, diverse energy- and QoS-aware power management schemes and strategies for wired communication networks to reduce energy consumption can be put forward by researchers. Speed scaling is available as DVFS on network processor units (NPUs), while sleep modes are available as DPM to decide when to sleep unneeded network components. These two dynamic power management techniques can be used either individually or cooperatively, and the distinct scheduling strategies based on them may influence the energy saving with different levels. Some results from power management literature were discussed in this chapter and will be used in the following chapters.

3.0 DVFS-based Power Management and QoS-aware Scheduling Strategies

The power management technique that we first focus on for wired communication networks is speed scaling. In current commercial routers, faster network processor units (NPUs) in line cards (LCs) can significantly improve network QoS performance. However, this improvement may come at a high cost of energy consumption. Using DVFS techniques, routers can adaptively adjust the operational frequencies of their processor units according to current network congestion. Excessive reduction in execution speeds to save energy, however, could result in QoS violation of the supported applications. To address the energy-QoS dichotomy, three families of DVFS-based, QoS-aware packet scheduling schemes are discussed in Chapter 3. The main objective of these proposed schemes is to sufficiently reduce the dynamic energy consumed by network routers through different speed scaling strategies, while achieving a fine balance between energy saving improvements and network performance requirements. Two metrics, namely queue length and link utilization, are considered to achieve this goal. Compared to existing methods under the same network environments, the results show that among our proposed speed scaling strategies, the highest energy saving can reach up to 10% dynamic energy saving of the total energy consumption, while also meeting the desired performance of the supported applications.

3.1 Introduction

The exponential growth of worldwide broadband subscribers, coupled with data- and compute-intensive applications that broadband deployment has enabled, is fueling the demand for higher Internet bandwidth to support the QoS requirements of these applications. An increase in bandwidth demand, however, comes at the costly price of higher power and energy consumption. Today's higher performance router LCs handle most data plane traffic processing tasks with specialized application-specific integrated circuit (ASIC) processors or other programmable hardware [Pierson, 2015; Group, 2007]. According to [Imaizumi and

Morikawa, 2010; Gupta and Singh, 2007a], LCs consume around 70% of the total router power, and the power consumption of NPUs accounts for more than 50% in each LC. Recent advances in semiconductor technology, which enabled higher parallelism and increased clock frequencies, paved the way to a new generation of power routers. These advances, however, come at a heavy price of increased power consumption, due to higher line card speeds [Pierson, 2015]. Therefore, seeking solutions to reducing power consumption, without adversely affecting network performance, becomes imperative for the design of future energy-efficient networks, with minimal impact on the environment.

Currently, two approaches are frequently used to manage power in computing and networking environments [Bolla et al., 2009; Etoh et al., 2008]. The first, referred to as *Speed Scaling*, uses dynamic voltage frequency scaling (DVFS) to control execution rates and reduce energy consumption [Bolla et al., 2009; Etoh et al., 2008; Bolla et al., 2011b; Tucker et al., 2009]. The second, referred to as *Dynamic Power Management (DPM)*, uses sleep mode to control power and save energy [Etoh et al., 2008; Bolla et al., 2011b; Chen et al., 2012; Yu et al., 2015b]. A large body of research work showed that DVFS can achieve significant power savings [Bolla et al., 2009; Etoh et al., 2008; Bolla et al., 2011b; Tucker et al., 2009]. However, excessive slowing-down of the processors may lead to an unacceptable level of QoS degradation of the supported applications. Consequently, dynamically adjusting processors' execution rates to reduce power and energy consumption while satisfying QoS requirements becomes a challenge.

To address this challenge, we propose and investigate three families of QoS-aware, DVFS-based packet scheduling schemes to control execution rates in line cards and reduce energy consumption. The major contributions of this chapter are: (i) the design of three families of QoS-aware, DVFS-based packet schedulers, based on link utilization, queue length and packet delay, respectively. Variants in each family, which differ in when and how decisions are made to adjust the execution rates, are derived; (ii) a holistic simulation framework, including an energy model, is proposed to investigate and compare the performance of each scheme, in different networking environments and traffic models; and (iii) a thorough performance study focusing on the energy consumption and network delay, for each scheduling scheme in each family, is carried out.

The rest of this chapter is organized by sections as follows: The related work is discussed in Section 3.2. The three families of DVFS-based, QoS-aware packet scheduling schemes and their variants are presented in Section 3.3, Section 3.4, Section 3.5 and Section 3.6. The simulation framework and the comparative analysis of the different QoS-aware strategies, under different network environments and traffic loads, are discussed in Section 3.7. Section 3.8 presents the conclusion of this chapter.

3.2 Related Work

Several energy-efficient schemes have been proposed for green networks [Bolla et al., 2009; Etoh et al., 2008; Puype et al., 2009]. Some of these schemes propose energy-based traffic engineering approaches designed to only keep a sufficient number of active routers, linecards, and interfaces to support the network workload. The remaining network devices are either shut down or put into sleep mode. Other research works focus on energy saving using DVFS-based power management approaches. In [Bolla et al., 2011b], Nedevschi, et al. present two simple power management algorithms, and explore the effect of sleep mode and DVFS-based rate adaptation on network energy saving. In [Tucker et al., 2009], Mandviwalla et al. propose three load-dependent strategies, i.e. Value Predictor (VP), Moving Average Predictor (MAP) and Exponentially-Weighted MAP (EWMAP), to reduce energy consumption in multiprocessor-based LCs. The results show that more than 60% dynamic power savings of the maximal dynamic power consumption can be achieved in one LC. Although these proposed schemes seek to reduce dynamic energy consumption at different levels through using link utilization in DVFS-enabled processors, they do not address the impact of the entire energy savings on QoS performance under different traffic loads in a network. On the other hand, considering the lack of a comprehensive router-based energy model, in [Group, 2007], Vishwanath, Arun, et al. propose a power model measurement methodology that quantifies the energy efficiency of high-capacity routing platforms at the packet- and byte-level. It focuses on network energy evaluation. The approach used to save energy, however, is not discussed and analyzed in detail. In this chapter, our proposed QoS-

aware, energy-minimizing packet schedulers address this shortcoming and seek a balanced tradeoff between network energy savings and acceptable levels of network QoS performance under different network environments and traffic loads, based on a derived comprehensive router-based energy model. This is achieved by controlling the NPU execution rates based on **queue length** or **link utilization**. Both metrics are critical to maintaining packet delay within acceptable levels of QoS performance [Yu et al., 2015a,b].

3.3 DVFS-Scheduler Design and Architecture

The DVFS-Scheduler dynamically adjusts processor frequency, based on the current state of the network, to reduce energy while meeting QoS performance. To design a QoS-aware, energy-efficient strategy for speed-scaling, several issues must be addressed. The first issue is related to monitoring network traffic to determine current network congestion. This information is used to adjust processor frequency, accordingly. When the congestion is high, the frequency must be scaled up, in order to meet the QoS requirements of the application. When the network congestion is low, however, the frequency is scaled down to reduce energy consumption, without violating QoS performance. The second issue deals with accurately determining the congestion granularity and time scale needed to effectively manage frequency scaling. A finer congestion granularity measured over a short time scale leads to higher accuracy, but at the expense of additional overhead. A tradeoff between granularity, time scale and overhead must, therefore, be worked out to achieve accuracy while maintaining a low overhead. The third issue deals with the scheduler's aggressivity when scaling the processor's frequency up or down. An aggressive strategy to lower processor speed to save energy, when the network congestion is low, may lead to a violation of QoS performance. Similarly, an aggressive strategy to increase processor speed in response to a high burst of traffic may lead to energy waste. The strategy must, therefore, achieve the right balance between saving energy and adhering to QoS performance.

To address the above issues, a DVFS-based scheduling architecture, depicted in Figure 3.1, is proposed. The architecture has three main components: Traffic Monitor (TM),

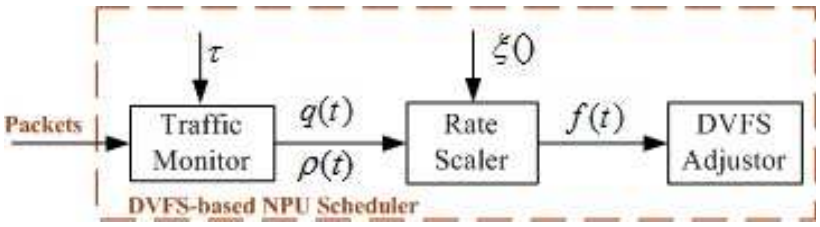

Figure 3.1: DVFS-Scheduler basic architecture.

Rate Scaler (RS) and DVFS Adjustor (DA). The TM component monitors the packets, over an interval τ, and compute the statics related to the state of the network. Depending on the scheduling strategy, the queue length, $q(\tau)$, or the link utilization, $\rho(\tau)$, are used to scale up or down the Network Processor Unit (NPU) execution rates. The RS component computes a network state-dependent scaling function, $\xi()$, which takes into consideration the aggressivity factor of the scheduling strategy, η, and the current level of network congestion. The scaling function is used by the DA component to adjust the NPU frequency, $f(\tau)$.

Table 3.1: Three families of DVFS-based, QoS-aware packet scheduling schemes.

Family	**Scheme**	**Metric**	**Scaling Factor**	**Scaling Strategy**
Load-aware [Yu et al., 2015a]	LA	Average traffic Load	ρ	$f(\tau_k) = max\,(f_{min}, min(f_{max} \cdot \rho(\tau_k), f_{max}))$
	$\bar{\text{L}}$A	Predicted traffic Load (EWMA)	$\overline{\rho}$	$f(\tau_k) = max\,(f_{min}, min(f_{max} \cdot \overline{\rho}(\tau_k), f_{max}))$
QL-aware [Yu et al., 2015a]	sQLA	Current Queue Length	$(\frac{q+1}{Q+1})^{\eta}$	$f(\tau_k) = max\left(f_{min}, f_{max} \cdot (\frac{q(t_k)+1}{Q+1})^{\eta}\right)$
	s$\bar{\text{Q}}$LA	Predicted Queue Length (EWMA[1])	$(\frac{\overline{q}+1}{Q+1})^{\eta}$	$f(\tau_k) = max\left(f_{min}, f_{max} \cdot (\frac{\overline{q}(\tau_k)+1}{Q+1})^{\eta}\right)$
	mQLA	Current Queue Length	$(\frac{(q-ql)+1}{(qh-ql)+1})^{\eta}$	$f(\tau_k) = \begin{cases} f_{min}, & \text{if } q(t_k) \leq ql \\ f_{max}, & \text{if } q(t_k) > qh \\ f_{min} + (f_{max} - f_{min}) \cdot (\frac{(q(t_k) - ql) + 1}{(qh - ql) + 1})^{\eta}, & \\ & \text{if } ql < q(t_k) \leq qh \end{cases}$
	m$\bar{\text{Q}}$LA	Predicted Queue Length (EWMA)	$(\frac{(\overline{q}-ql)+1}{(qh-ql)+1})^{\eta}$	$f(\tau_k) = \begin{cases} f_{min}, & \text{if } \overline{q}(\tau_k) \leq ql \\ f_{max}, & \text{if } \overline{q}(\tau_k) > qh \\ f_{min} + (f_{max} - f_{min}) \cdot (\frac{(\overline{q}(\tau_k) - ql) + 1}{(qh - ql) + 1})^{\eta}, & \\ & \text{if } ql < \overline{q}(\tau_k) \leq qh \end{cases}$
Delay-aware [Yu et al., 2015b]	QLDA	Predicted Queue Length & Packet Delay (EWMA & GPR[2])	$(\frac{\overline{q}-ql}{qh-ql})^{\eta(\rho)}$	$f(\tau_k) = \begin{cases} f_{min}, & \text{if } \overline{q}(\tau_k) \leq q_l \\ f_{max}, & \text{if } \overline{q}(\tau_k) > q_h \\ f(\tau_{k-1}), & \\ & \text{if } q_l < \overline{q}(\tau_k) \leq q_h \text{ and } \left\lvert\overline{d}(\tau_k) - d^T\right\rvert \leq \tilde{dv}(\tau_k) \\ f_{min} + (f_{max} - f_{min}) \cdot (\frac{\overline{q}(\tau_k) - ql}{qh - ql})^{\eta(\rho(\tau_k))}, & \\ & \text{if } q_l < \overline{q}(\tau_k) \leq q_h \text{ and } \left\lvert\overline{d}(\tau_k) - d^T\right\rvert > \tilde{dv}(\tau_k) \end{cases}$

Motivated by our research work on DVFS-based dynamic power management, we explored and developed three families of DVFS-based, QoS-aware packet scheduling schemes to reduce the energy consumption of routers based on different strategies, operating under different traffic loads and network topologies under the same QoS constraints, as summarized in Table 3.1. Using DVFS, routers can adaptively adjust the operational frequencies of their processors, based on current conditions of the network. In these schemes, two metrics, namely current queue length and link utilization, are considered.

The first family of schemes studied in this dissertation uses the network traffic load, i.e. link utilization, to dynamically scale the processor's speed [Pierson, 2015]. The first Load-based scheme, referred to as Load-aware scheduler (LA), uses the traffic load over packet inter departure interval to adjust the NPU frequency. The second scheme, referred to as predicted Load-aware scheduler ($\bar{\text{L}}$A), uses the predicted average load by exponentially weighted moving average (EWMA) algorithm over a given interval time, to adjust the frequency. In this family, when the load is high, the speed is scaled up, and when the load is low, the speed is scaled down [Yu et al., 2015a]. The detail is introduced in Section 3.4.

The second family of schemes, referred to as Single-threshold, QL-aware scheduler (sQLA), Multi-threshold, QL-aware Scheduler (mQLA), Single-threshold Average QL-aware Scheduler (s$\bar{\text{Q}}$LA) and Multi-threshold Average QL-aware Scheduler (m$\bar{\text{Q}}$LA) [Pierson, 2015], uses the queue length, instead of link utilization, to scale the frequency [Yu et al., 2015a]. When the queue length increases, the scheduler scales the frequency up to meet the QoS delay requirements. When the queue length decreases, however, the scheduler scales the frequency down to save energy, without violating the QoS requirements. sQLA and mQLA use instantaneous queue length when adjusting NPU frequency. As such, they are instantly responsive to traffic load variation. Insights derived from these schemes are valuable in gaining a better understanding of the energy saving levels that can be achieved. These schemes, however, are not feasible in real networks, due to the overhead caused by excessive frequency adjustments. A practical implementation of these schemes can be achieved by using the average, as opposed to the instantaneous, queue length. Therefore, the resulting average queue-length based schemes, s$\bar{\text{Q}}$LA and m$\bar{\text{Q}}$LA, both use the EWMA algorithm to estimate the queue length periodically over a given interval to dynamically adjust processor frequencies [Yu

et al., 2015a]. The detail is introduced in Section 3.5.

The third family scheme, referred to as Queue Length (QL)-based, Delay-aware packet scheduler (QLDA) [Yu et al., 2015b], is an extended scheme of m$\bar{\text{Q}}$LA [Yu et al., 2015a]. In this scheme, a DVFS-based, Delay-aware scheduler is proposed, to decide when and how to adjust the router execution rates are based on not only the predicted queue-length but also the target packet delay. The goal is to scale network processor frequency and achieve maximal energy saving, under QoS delay requirements [Yu et al., 2015b]. The detail is introduced in Section 3.6.

3.4 Load-aware DVFS-Schedulers

Load-aware DVFS adjusts the NPU frequency based on link utilization. To this end, the scheduler increases the NPU frequency to meet QoS requirements when the load increases, and decreases it to save energy when the load decreases. The effectiveness of the Load-aware DVFS-Schedulers depends on what levels of load granularity are used in the scheduling decision when the load is measured, and how aggressive is the scheduler in its quest to reduce energy. In [Yu et al., 2015a], we propose two Load-aware schedulers and investigate their performance, based on the load granularity and the scheduler's aggressiveness toward energy saving.

3.4.1 Load-aware Scheduler (LA)

The LA scheme uses the link utilization, upon the departure of a packet, to adjust dynamically the processor frequency. Similarly, let τ_k be the elapsed time interval upon the departure of packet k, $k \geq 1$. The link utilization, $\rho(\tau_k)$, is defined as the ratio of the packet arrival rate, $\lambda(\tau_k)$, to the packet service rate, $\mu(\tau_k)$, over the time interval, τ_k. Depending on the traffic burstiness and the level of network congestion, the NPU load during the service of a packet may either decrease or increase. The RS determines $\rho(\tau_k)$ as the Load-based scaling function, $\xi_\rho()$, over the k^{th} packet inter departure time, and scales the NPU

execution frequency, either up or down, based on Eq. 3.1. Note that, the adjusted frequency must not exceed the maximum frequency, f_{max}, and must not be less than the minimum frequency, f_{min}.

$$f(\tau_k) \;=\; max\left(f_{min}, min(f_{max} \cdot \xi_\rho(\tau_k), f_{max})\right) \tag{3.1}$$

Let $\Delta A(\tau_k)$ and $\Delta D(\tau_k)$ denote the number of packet arrivals and departures over the interval τ_k, respectively. The Load-based scaling factor, $\xi_\rho(\tau_k)$, can be computed as follows:

$$\xi_\rho(\tau_k) = \frac{\lambda(\tau_k)}{\mu(\tau_k)} = \frac{\Delta A(\tau_k)}{\Delta D(\tau_k)} \tag{3.2}$$

$\xi_\rho(\tau_k)$ is the Load-based scaling factor used to adjust the NPU execution frequency, taking into consideration the dynamics of the network congestion level, while seeking to minimize energy consumption. The basic steps of the LA scheduling scheme are described in Algorithm 3.1.

Algorithm 3.1 LA Scheduling Scheme.

1: **For each NPU in LC at the router**
2: **Initialization:**
3: $A(0), D(0) \leftarrow 0, f(\tau_0) \leftarrow f_{Inital}$
4: **Measure link utilization** $\rho(\tau_k)$ **at** k^{th} **packet departure time**
5: $\tau_k \leftarrow$ **the** k^{th} **inter departure time**
6: **Calculate average arrival rate**
7: $\Delta A(\tau_k) \leftarrow A(k) - A(k-1), \lambda(\tau_k) \leftarrow \frac{\Delta A(\tau_k)}{\tau_k}$
8: **Calculate departure rate**
9: $\Delta D(\tau_k) \leftarrow D(k) - D(k-1), \mu(\tau_k) \leftarrow \frac{\Delta D(\tau_k)}{\tau_k}$
10: **Calculate the link utilization**
11: $\rho(\tau_k) \leftarrow \frac{\lambda(\tau_k)}{\mu(\tau_k)}$
12: **Calculate scaling factor,** $\xi_\rho(\tau_k)$
13: $\xi_\rho(\tau_k) \leftarrow \rho(\tau_k)$
14: **Scale frequency,** $f(\tau_k)$
15: $f(\tau_k) \leftarrow max\left(f_{min}, min(f_{max} \cdot \xi_\rho(\tau_k), f_{max})\right)$

Saved Variables:
- $A(k)$: the accumulated arrival packets until the k^{th} packet departure time.
- $D(k)$: the accumulated departure packets until the k^{th} packet departure time.

3.4.2 Predicted Load-aware Scheduler ($\bar{\text{LA}}$)

Contrary to the LA scheduler, the $\bar{\text{LA}}$ scheduler uses the predicted load, $\overline{\rho}(\tau_k)$ over a given time interval, τ, to adjust the NPU execution speed. To this end, it uses the EWMA algorithm to predict the average packet arrival rate, $\overline{\lambda}(\tau_k)$, as shown in Eq. 3.3, over the k^{th} time interval τ, $k \geq 1$.

$$\overline{\lambda}(\tau_k) = (1 - w_a(\tau_k)) \cdot \overline{\lambda}(\tau_{k-1}) + w_a(\tau_k) \cdot \lambda(\tau_k) \tag{3.3}$$

In the above equation, $\lambda(\tau_k)$ represents the packet arrival rate over the time interval τ_k, and $w_a(\tau_k)$ is the traffic weight factor defined as:

$$w_a(\tau_k) = c_a \cdot \frac{e_a^2(\tau_k)}{\sigma_a(\tau_k)} \tag{3.4}$$

The term $e_a(\tau_k)$ represents the load prediction error function, defined as $e_a(\tau_k) = \lambda(\tau_k) - \overline{\lambda}(\tau_k)$. The term $\sigma_a(\tau_k)$ denotes the square prediction error for the time interval τ_k, defined as $\sigma_a(\tau_k) = c_a \cdot e_a^2(\tau_k) + (1 - c_a) \cdot \sigma(\tau_{k-1})$, where c_a is a constant parameter within $(0, 1)$, which is used to estimate to the value of $\sigma_a(\tau_k)$. The first order auto-regressive filter used to predict future traffic load, combined with the error prediction method used to adaptively compute the smooth factor $w_a(\tau_k)$, guarantee that the predicted traffic load is not affected by small deviations.

The $\bar{\text{LA}}$ scheme also use the above Eq. 3.2 to compute its scaling factor, $\xi_{\overline{\rho}}(\tau_k)$. In the LA scheme, the number of packet departures over two consecutive packet departures is always equal to 1, while in the $\bar{\text{LA}}$ scheme, τ_k, $k \geq 1$, refers to a generic, regular time interval τ, during which more packet departures may occur. The basic steps of the $\bar{\text{LA}}$ scheduling scheme are described in Algorithm 3.2.

Algorithm 3.2 $\bar{\text{L}}$A Scheduling Scheme.

1: **For each NPU in LC at the router**
2: **Initialization:**
3: $A(0), D(0), \overline{\lambda}(\tau_0) \leftarrow 0, k \leftarrow 1, f(\tau_0) \leftarrow f_{Inital}$
4: **Monitoring traffic load over the k^{th} interval τ**
5: **Calculate average arrival rate**
6: $\Delta A(\tau_k) \leftarrow A(k) - A(k-1), \lambda(\tau_k) \leftarrow \frac{\Delta A(\tau_k)}{\tau}$
7: **Calculate average departure rate**
8: $\Delta D(\tau_k) \leftarrow D(k) - D(k-1), \mu(\tau_k) \leftarrow \frac{\Delta D(\tau_k)}{\tau}$
9: **Update the dynamic smooth filter $w_a(\tau_k)$ with Eq. 3.4**
10: **Predict the new link utilization $\overline{\rho}(\tau_k)$ for the k^{th} interval τ**
11: $\overline{\lambda}(\tau_k) \leftarrow (1 - w_a(\tau_k)) \cdot \overline{\lambda}(\tau_{k-1}) + w_a(\tau_k) \cdot \lambda(\tau_k)$
12: $\overline{\rho}(\tau_k) \leftarrow \frac{\overline{\lambda}(\tau_k)}{\mu(\tau_k)}$
13: **Calculate scaling factor, $\xi_{\overline{\rho}}(\tau_k)$**
14: $\xi_{\overline{\rho}}(\tau_k) \leftarrow \overline{\rho}(\tau_k)$
15: **Scale frequency, $f(\tau_k)$**
16: $f(\tau_k) \leftarrow max\left(f_{min}, min(f_{max} \cdot \xi_{\overline{\rho}}(\tau_k), f_{max})\right)$
17: $k \leftarrow k + 1$

Saved Variables:
- $A(k)$: the accumulated arrival packets until the end of the k^{th} interval τ.
- $D(k)$: the accumulated departure packets until the end of the k^{th} interval τ.

3.5 QL-aware DVFS-Schedulers

In contrast to Load-aware DVFS, the basic QL-aware DVFS-Scheduler seeks to reduce energy, while adhering to QoS performance, by dynamically adjusting processor frequency, based on queue length derived metrics. Different variants of this scheme can be designed, depending on the metric and the level of granularity used to characterize network congestion, and the aggressivity of the scheme to achieve higher energy saving.

Two metrics, namely *instantaneous* and *average* queue length, are used to characterize congestion. Based on these metrics, four QL-aware DVFS-Schedulers, using single and multiple queue length thresholds, are proposed. These schedulers differ in the strategy used

to account for queue length and the method used to capture different granularity and time scales of the network congestion.

3.5.1 Single-threshold, QL-aware Scheduler (sQLA)

The sQLA scheme uses the queue length, $q(t_k)$, upon the k^{th} packet departure time, t_k, $k \geq 1$, to adjust dynamically the NPU execution frequency. Let τ_k be the elapsed departure interval upon the departure of packet k, $k \geq 1$. The frequency, $f(\tau_k)$, over τ_k, is defined in Eq. 3.5.

$$f(\tau_k) = max\left(f_{min}, f_{max} \cdot \xi_q\left(\tau_k\right)\right) \tag{3.5}$$

The scaling function, $\xi_q()$, is determined based on the queue occupancy, defined as the ratio of the queue length, $q(t_k)$, to the maximum queue capacity, Q, raised to the power, η.

$$\xi_q(\tau_k) = (\frac{q(t_k)+1}{Q+1})^{\eta} \tag{3.6}$$

The scaling function, $(0 < \xi_q() \leq 1)$, adapts the NPU frequency to the current queue length. As the queue length increases, the sQLA scheduler scales the frequency up to meet the QoS delay requirements. When the queue length decreases, however, the sQLA scheduler scales the frequency down to save energy, without violating the QoS requirements. Algorithm 3.3 describes the basic steps of the sQLA scheduling scheme.

Algorithm 3.3 sQLA Scheduling Scheme.

1: **For each NPU in LC at the router**
2: **Initialization:**
3: $f(\tau_0) \leftarrow f_{Inital}$
4: **Measure current queue length,** $q(t_k)$**, at the** k^{th} **packet departure time**.
5: **Calculate scaling factor,** $\xi_q(\tau_k)$
6: $\xi_q(\tau_k) \leftarrow (\frac{q(t_k)+1}{Q+1})^{\eta}$
7: **Scale frequency,** $f(\tau_k)$
8: $f(\tau_k) \leftarrow max\left(f_{min}, f_{max} \cdot \xi_q(\tau_k)\right)$

sQLA relies exclusively on queue length to schedule packets. As such, it can easily be incorporated in packet scheduling schemes commonly used in current routers, such as FIFO,

priority-based, and weighted fair queuing. By adjusting NPU frequency at the packet level, the scheme has the potential to lead to significant energy savings. Its main shortcoming, however, is its inability to capture finer levels of congestion granularity, when controlling NPU execution rates. More specifically, coarse congestion granularity may underestimate the current traffic load, which, in turn, may lead to violations of QoS requirements. Overestimating current traffic load, on the other hand, may lead to a missed opportunity to reduce energy. To address this shortcoming, we introduce a multi-threshold variant of the sQLA scheme.

3.5.2 Multi-threshold, QL-aware Scheduler (mQLA)

Similar to sQLA, mQLA uses queue length, upon the departure of a packet, to adjust NPU execution rates. Contrary to sQLA, however, mQLA uses a coarser level of network congestion granularity in its decision to scale up or down the NPU execution rates. More specifically, mQLA uses two queue length thresholds, namely q_l and q_h ($0 \leq q_l < q_h \leq Q$), to specify three network congestion regions, namely *low, medium* and *high*, as depicted in Figure 3.2.

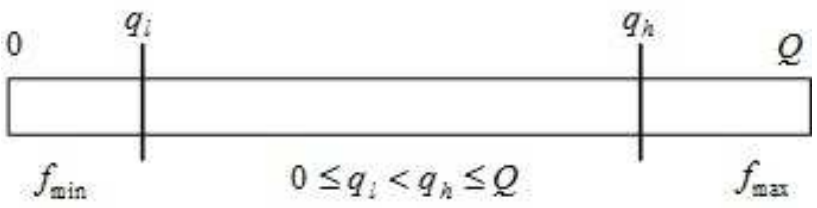

Figure 3.2: Packet buffer.

mQLA depends on a scaling function based on these three packet buffer occupancy regions, as illustrated in Eq. 3.7.

$$f(\tau_k) = \begin{cases} f_{min}, & \text{if } q(t_k) \leq q_l \\ f_{max}, & \text{if } q(t_k) > q_h \\ f_{min} + (f_{max} - f_{min}) \cdot \xi_q^{'}(\tau_k), & \text{if } q_l < q(t_k) \leq q_h \end{cases} \tag{3.7}$$

In the above Equation, the scaling factor is defined as Eq. 3.8.

$$\xi_q^{'}(\tau_k) = \left(\frac{(q(t_k) - q_l) + 1}{(q_h - q_l) + 1} \right)^{\eta}, \text{ if } q_l < q(t_k) \leq q_h \tag{3.8}$$

Algorithm 3.4 describes the basic steps of the mQLA scheduling scheme.

Algorithm 3.4 mQLA Scheduling Scheme.

For each NPU in LC at the router
Initialization:
$f(\tau_0) \leftarrow f_{Inital}$
Measure current queue length, $q(t_k)$, at the k^{th} packet departure time.
if $q(t_k) \leq q_l$ **then**
    Scale frequency $f(\tau_k)$ **to** f_{min}
    $f(\tau_k) \leftarrow f_{min}$
else if $q(t_k) > q_h$ **then**
    Scale frequency $f(\tau_k)$ **to** f_{max}
    $f(\tau_k) \leftarrow f_{max}$
else
    Calculate scaling factor, $\xi_q^{'}(\tau_k)$
    $\xi_q^{'}(\tau_k) \leftarrow (\frac{(q(t_k)-q_l)+1}{(q_h-q_l)+1})^{\eta}$
    Scale frequency, $f(\tau_k)$
    $f(\tau_k) \leftarrow f_{min} + (f_{max} - f_{min}) \cdot \xi_q^{'}(\tau_k)$
end if

The above schemes use instantaneous queue length when adjusting NPU frequency. As such, they are instantly responsive to traffic load variation. Insights derived from these schemes are valuable in gaining a better understanding of the energy saving levels that can be achieved. These schemes, however, are not feasible in real networks, due to the overhead caused by excessive frequency adjustments. A practical implementation of these schemes can be achieved by using the average, as opposed to the instantaneous, queue length. These schemes are described next.

3.5.3 Single-threshold Average QL-aware Scheduler (s$\bar{\text{Q}}$LA)

The s$\bar{\text{Q}}$LA scheme uses the exponentially weighted moving average (EWMA) algorithm to periodically predict the average queue length over a given time interval, τ. Consequently, the average queue length, $\overline{q}(\tau_k)$, for the k^{th} time interval, τ_k, $k \geq 1$, is defined as:

$$\overline{q}(\tau_k) = (1 - w_q(\tau_k)) \cdot \overline{q}(\tau_{k-1}) + w_q(\tau_k) \cdot q(t_k) \tag{3.9}$$

In the above Eq. 3.9, $q(t_k)$ represents the queue length measured at the ending time of the k^{th} time interval, t_k, and $w_q(\tau_k)$ is the queue-length weight factor defined as:

$$w_q(\tau_k) = c_q \cdot \frac{e_q(\tau_k)^2}{\sigma_q(\tau_k)} \tag{3.10}$$

The term $e_q(\tau_k)$ represents the queue length prediction error function, defined as $e_q(\tau_k) = q(t_k) - \overline{q}(\tau_k)$, and $\sigma_q(\tau_k)$ denotes the square prediction error for the interval τ_k, defined as $\sigma_q(\tau_k) = c_q \cdot e_q^2(\tau_k) + (1 - c_q) \cdot \sigma_q(\tau_{k-1})$, where c_q is a constant parameter within $(0, 1)$, which is used to estimate the value of $\sigma_q(\tau_k)$. The first order auto-regressive filter used to predict future queue length, combined with the error prediction method used to adaptively compute the weight function $w_q(\tau_k)$, guarantee that the predicted queue length is not affected by small deviations. Algorithm 3.5 describes the basic steps of the s$\bar{\text{Q}}$LA scheduling scheme.

Algorithm 3.5 s$\bar{\text{Q}}$LA Scheduling Scheme.

1: **For each NPU in LC at the router**
2: **Initialization:**
3: $\overline{q}(\tau_0) \leftarrow 0, k \leftarrow 1, f(\tau_0) \leftarrow f_{Inital}$
4: **Measure queue length,** $q(t_k)$**, at the ending time of the** k^{th} **interval** τ
5: **Update the dynamic smooth filter** $w_q(\tau_k)$ **with Eq. 3.10**
6: **Estimate the new average** $\overline{q}(\tau_k)$ **for the** k^{th} **interval** τ
7: $\overline{q}(\tau_k) \leftarrow (1 - w_q(\tau_k)) \cdot \overline{q}(\tau_{k-1}) + w_q(\tau_k) \cdot q(t_k)$
8: **Calculate scaling factor,** $\xi_{\overline{q}(\tau_k)}$
9: $\xi_{\overline{q}}(\tau_k) \leftarrow (\frac{\overline{q}(\tau_k)+1}{Q+1})^{\eta}$
10: **Scale frequency,** $f(\tau_k)$
11: $f(\tau_k) \leftarrow max\,(f_{min}, f_{max} \cdot \xi_{\overline{q}}(\tau_k))$
12: $k \leftarrow k + 1$

3.5.4 Multi-threshold Average QL-aware Scheduler (m$\bar{\text{Q}}$LA)

Similarly, as a variant of the mQLA scheme, the m$\bar{\text{Q}}$LA scheme uses the average queue length estimated by EWMA method and multi-threshold strategy to adaptively control the execution rates of line cards. Algorithm 3.6 describes the basic steps of the m$\bar{\text{Q}}$LA scheduling scheme.

Algorithm 3.6 m$\bar{\text{Q}}$LA Scheduling Scheme.

For each NPU in LC at the router
Initialization:
$\overline{q}(\tau_0) \leftarrow 0, k \leftarrow 1, f(\tau_0) \leftarrow f_{Inital}$
Measure queue length, $q(t_k)$, at the end time of the k^{th} interval τ
Update the dynamic smooth filter $w_q(\tau_k)$ with Eq. 3.10
Estimate the new average $\overline{q}(\tau_k)$ for the k^{th} interval τ
$\overline{q}(\tau_k) \leftarrow (1 - w_q(\tau_k)) \cdot \overline{q}(\tau_{k-1}) + w_q(\tau_k) \cdot q(t_k)$
if $\overline{q}(\tau_k) \leq q_l$ **then**
    Scale frequency $f(\tau_k)$ to f_{min}
    $f(\tau_k) \leftarrow f_{min}$
else if $\overline{q}(\tau_k) > q_h$ **then**
    Scale frequency $f(\tau_k)$ to f_{max}
    $f(\tau_k) \leftarrow f_{max}$
else
    Calculate scaling factor, $\xi'_{\overline{q}}(\tau_k)$
    $\xi'_{\overline{q}}(\tau_k) \leftarrow (\frac{(\overline{q}(\tau_k) - q_l) + 1}{(q_h - q_l) + 1})^{\eta}$
    Scale frequency, $f(\tau_k)$
    $f(\tau_k) \leftarrow f_{min} + (f_{max} - f_{min}) \cdot \xi'_{\overline{q}}(\tau_k)$
end if
$k \leftarrow k + 1$

3.6 Delay-aware DVFS-Scheduler

Queue Length (QL)-based, Delay-aware packet scheduler (QLDA) [Yu et al., 2015b], is an extended scheme of m$\bar{\text{Q}}$LA. It uses multiple queue length thresholds to accurately capture

network congestion. In response to different levels of network congestion, different NPU-rate scaling strategies are used to determine when and how NPU execution rates are adjusted based on the predicted queue length and the estimated delay variance, aiming to achieve adequate energy savings under different traffic loads, without degrading delay performance.

3.6.1 Delay-aware DVFS-Scheduler Design and Architecture

The basic idea of DVFS-Scheduler is to dynamically adjust the processor frequency, based on the current state of the network, to reduce energy consumption. Similarly, to design an effective QL-based Delay-aware DVFS-Scheduler, several issues must be addressed. First, a strategy must be in place to determine how queue length impacts scheduling decisions. Second, appropriate levels of congestion granularity must be taken into consideration when adjusting the NPU's execution rate. Multiple queue length thresholds to model different levels of network congestion are considered in this chapter. Third, a mechanism must be in place to predict traffic end-to-end delay and bind its variability so that the desired delay performance can be achieved. In the proposed scheduler, an effective method, which estimates packet delay variability to predict the deviation of packet delay from the target end-to-end delay, is used to control NPU's execution rate adjustments. Finally, an adaptive mechanism must be in place to control the "aggressivity" of the scheduling policy to ensure energy savings without degrading QoS performance.

To address the above issues, a Delay-aware DVFS-enabled scheduling architecture, depicted in Figure 3.3, is proposed. The Traffic Monitor (TM) monitors the packet queue and gathers statics related to its length. The estimated average queue length, $\overline{q}(\tau)$, and average packet delay, $\overline{d}(\tau)$, over a time interval τ, are used to scale up or down the NPU execution rates. The NPU Rate Scaler (RS) computes a network state-dependent scaling function, $\xi()$, taking into consideration the aggressivity factor of the scheduling strategy, $\eta(\tau)$ generated by the average network traffic load $\rho(\tau)$, and the current level of network congestion. The DVFS Adjustor (DA) adjusts the NPU frequency, $f(\tau)$, based on the scaling factor.

Based on the above architecture, a Delay-aware DVFS-enabled packet scheduler is proposed, which uses predicted average queue length and average packet delay to control the

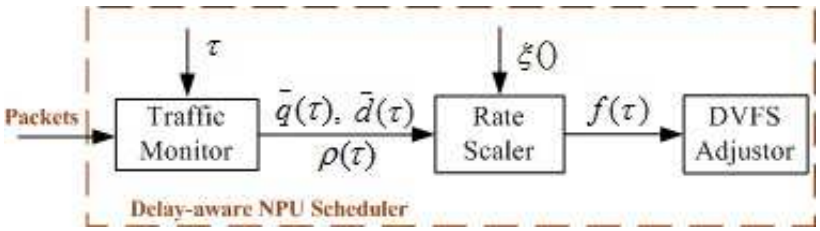

Figure 3.3: Delay-aware DVFS-enabled scheduling architecture.

frequency adjustment. The related NPU rate scaling strategies, queue-length, and packet-delay prediction mechanisms are introduced in the following.

3.6.2 QL-based Delay-Aware Packet Scheduler (QLDA)

Similarly, QLDA uses the exponentially weighted moving average (EWMA) scheme to periodically predict the average queue length over a given time interval, τ, to adjust the NPU execution frequency dynamically. In order to reduce DVFS switching overhead, QLDA uses a coarser level of network congestion granularity in its decision to scale up or down the NPU execution rates. More specifically, QLDA uses two queue length thresholds, namely q_l and q_h $(0 \leq q_l < q_h \leq Q)$ to define *low, medium* and *high* network congestion regions, as depicted in Figure 3.2.

According to the above three packet buffer occupancy regions Fig.3.2, the frequency, $f(\tau_k)$, over the k^{th} time interval, τ_k, $k \geq 1$, is defined in Eq. 3.11.

$$f(\tau_k) = \begin{cases} f_{min}, & \text{if } \overline{q}(\tau_k) \leq q_l \\ f_{max}, & \text{if } \overline{q}(\tau_k) > q_h \\ f(\tau_{k-1}), & \text{if } q_l < \overline{q}(\tau_k) \leq q_h \text{ and } \left|\overline{d}(\tau_k) - d^T\right| \leq \tilde{dv}(\tau_k) \\ f_{min} + (f_{max} - f_{min}) \cdot \xi(\tau_k), & \\ & \text{if } q_l < \overline{q}(\tau_k) \leq q_h \text{ and } \left|\overline{d}(\tau_k) - d^T\right| > \tilde{dv}(\tau_k) \end{cases} \tag{3.11}$$

In the above strategy function, the scaling factor, $\xi(\tau_k)$, over τ_k, as illustrated in Eq. 3.12, is determined based on the queue occupancy in the middle region (q_l, q_h), defined as the ratio

of the queue length occupancy to the difference between two queue length thresholds, raised to the power, $\eta(\rho)$.

$$\xi(\tau_k) = (\frac{\overline{q}(\tau_k) - q_l}{q_h - q_l})^{\eta(\rho)} \tag{3.12}$$

The average queue length, $\overline{q}(\tau_k)$, over τ_k, is defined in Eq. 3.13. $q(\tau_k)$ represents the queue length at the end of the interval τ_k, and $w^q(\tau_k)$, defined as $w^q(\tau_k) = c_q \cdot \frac{eq^2(\tau_k)}{\sigma q(\tau_k)}$, where $0 < c_q < 1$, $0 < w^q(\tau_k) < 1$, is the queue-length weight factor.

$$\overline{q}(\tau_k) = (1 - w^q(\tau_k)) \cdot \overline{q}(\tau_{k-1}) + w^q(\tau_k) \cdot q(\tau_k) \tag{3.13}$$

The term $eq(\tau_k)$ represents the queue length prediction error function, defined as $eq(\tau_k) = q(\tau_k) - \overline{q}(\tau_k)$, and $\sigma q(\tau_k)$ denotes the square prediction error for τ_k, defined as $\sigma q(\tau_k) = c_q \cdot eq^2(\tau_k) + (1 - c_q) \cdot \sigma q(\tau_{k-1})$. The first order auto-regressive filter used to predict future queue length, combined with the error prediction method used to adaptively compute the weight function, $w^q(\tau_k)$, guarantee that the predicted queue length is not affected by small deviations.

The aggressivity factor $\eta(\rho(\tau_k))$ associated with the average traffic load $\rho(\tau_k)$, over τ_k, is defined as Eq. 3.14.

$$\eta(\rho(\tau_k)) = a \cdot e^{-(\frac{\rho(\tau_k) - b}{c})^2} \tag{3.14}$$

The aggressivity function, $\eta()$, uses Gaussian regression model to generate the aggressivity factor of the scheduling strategy based on the traffic load. a, b, and c are constant model parameters. The average traffic load $\rho(\tau_k)$ is computed by the traffic average arrival rate $\overline{\lambda}(\tau_k)$ over τ_k and the maximal NPU service rate u_{max}, as illustrated Eq. 3.15.

$$\rho(\tau_k) = \frac{\overline{\lambda}(\tau_k)}{u_{max}} \tag{3.15}$$

In order to further reduce DVFS scaling overhead, QLDA uses the estimated delay variance to decide when to adjust NPU's frequency. When the estimated average queue length falls in the middle region (q_l, q_h), QLDA does not systematically scale up or down the NPU's execution rate over every time interval τ. DVFS scaling in this region only takes place when

the absolute value of the deviation between the predicted average packet delay, $\overline{d}(\tau_k)$, over τ_k, and the target packet delay, d^T, exceeds the estimated delay deviation, $\tilde{dv}(\tau_k)$. When a deviation occurs, the frequency is scaled up or down, depending on the queue length and the target packet delay, as illustrated in Eq. 3.11.

In order to predict the average packet delay, $\overline{d}(\tau_k)$, the same exponential smoothing technique, defined in Eq. 3.16, is used. $d(\tau_k)$ represents the k^{th} average packet delay, which is computed based on the average queue length, the average arrival rate and the average departure rate over τ_k.

$$\overline{d}(\tau_k) = (1 - w^d(\tau_k)) \cdot \overline{d}(\tau_{k-1}) + w^d(\tau_k) \cdot d(\tau_k) \tag{3.16}$$

The term $w^d(\tau_k)$, $0 < w^d(\tau_k) < 1$, denotes a dynamic delay weight factor and is defined as $w^d(\tau_k) = c_d \cdot \frac{ed^2(\tau_k)}{\sigma d(\tau_k)}$, where $0 < c_d < 1$. Similarly, $ed(\tau_k)$ represents the delay prediction error function, defined as $ed(\tau_k) = d(\tau_k) - \overline{d}(\tau_k)$, and $\sigma d(\tau_k)$ denotes the square prediction error for τ_k, defined as $\sigma d(\tau_k) = c_d \cdot ed^2(\tau_k) + (1 - c_d) \cdot \sigma d(\tau_{k-1})$. The first order auto-regressive filter used to predict future packet delay, combined with the error prediction method used to adaptively compute the weight function $w^d(\tau_k)$, guarantee that the predicted delay is not affected by small delay deviations.

Based on the delay prediction error function and a constant α_{dv}, $0 < \alpha_{dv} < 1$, ($\alpha_{dv} = 0.25$ is recommended), the delay deviation can be estimated in Eq. 3.17.

$$\tilde{dv}(\tau_k) = (1 - \alpha_{dv}) \cdot \tilde{dv}(\tau_{k-1}) + \alpha_{dv} \cdot |ed(\tau_k)| \tag{3.17}$$

QLDA relies exclusively on queue length to schedule packets. As such, it can easily be incorporated in packet scheduling schemes commonly used in current routers, such as FIFO, priority-based, and weighted fair queuing. Algorithm 3.7 describes the basic steps of the QLDA scheduling scheme.

Algorithm 3.7 QLDA Scheduling Scheme.

For each NPU in LC at the router
Initialization:
$\overline{q}(\tau_0), \overline{d}(\tau_0), \tilde{dv}(\tau_0) \leftarrow 0, k \leftarrow 1, f(\tau_0) \leftarrow f_{Inital}$
Monitor queue length, $q(\tau_k)$**, at the end time of** τ_k
Update the queue-length smooth filter, $w^q(\tau_k)$
Estimate the new average $\overline{q}(\tau_k)$ **for** τ_k
$\overline{q}(\tau_k) \leftarrow (1 - w^q(\tau_k)) \cdot \overline{q}(\tau_{k-1}) + w^q(\tau_k) \cdot q(\tau_k)$
Calculate $d(\tau_k)$ **based on** $\overline{q}(\tau_k)$
Estimate the new $\overline{d}(\tau_k)$ **and** $\tilde{dv}(\tau_k)$ **for** τ_k
Update the delay smooth filter, $w^d(\tau_k)$
$\overline{d}(\tau_k) \leftarrow (1 - w^d(\tau_k)) \cdot \overline{d}(\tau_{k-1}) + w^d(\tau_k) \cdot d(\tau_k)$
$ed(\tau_k) \leftarrow d(\tau_k) - \overline{d}(\tau_k)$
$\tilde{dv}(\tau_k) \leftarrow (1 - \alpha_{dv}) \cdot \tilde{dv}(\tau_{k-1}) + \alpha_{dv} \cdot |ed(\tau_k)|$
if $\overline{q}(\tau_k) \leq q_l$ **then**
    Scale frequency $f(\tau_k)$ **to** f_{min}
    $f(\tau_k) \leftarrow f_{min}$
else if $\overline{q}(\tau_k) \geq q_h$ **then**
    Scale frequency $f(\tau_k)$ **to** f_{max}
    $f(\tau_k) \leftarrow f_{max}$
else
    if $\left|\overline{d}(\tau_k) - d^T\right| \leq \tilde{dv}(\tau_k)$ **then**
        $f(\tau_k) \leftarrow f(\tau_{k-1})$
    else
        $\overline{\lambda}(\tau_k) \leftarrow A(\tau_k)/\tau_k$
        $\rho(\tau_k) \leftarrow \overline{\lambda}(\tau_k)/u_{max}$
        Generate aggressivity factor, $\eta(\rho(\tau_k))$
        Calculate scaling factor, $\xi(\tau_k)$
        $\xi(\tau_k) \leftarrow (\frac{\overline{q}(\tau_k) - q_l}{q_h - q_l})^{\eta(\rho(\tau_k))}$
        Scale frequency, $f(\tau_k)$
        $f(\tau_k) \leftarrow f_{min} + (f_{max} - f_{min}) \cdot \xi(\tau_k)$
    end if
end if
$k \leftarrow k + 1$

Parameters:
- u_{max}: a fixed parameter donates the maximal service rate at NPU.
- $A(\tau_k)$: a saved variable donates the number of the arrival packets over τ_k.

3.7 Evaluation

A simulation framework to assess the performance of the energy- and QoS-aware scheduling schemes discussed above are proposed in [Yu et al., 2015a,b]. We consider a set of DVFS-enabled routers and present a detailed model to determine the *packet-based* and *router-based* energy consumption, taking into consideration the frequency adjustment strategy used by the underlying scheduler. A NS2-based simulation framework is used to assess the performance of the proposed strategies in terms of energy gain.

3.7.1 Packet- and Router-based Energy Consumption Models

Two main components impact power consumption in network routers [Vishwanath Member et al., 2014; Valentini et al., 2013]. The first, referred to as static power, arises from the bias and leakage current to support control plane, environment units, and load-independent data plane [Vishwanath Member et al., 2014]. The second, referred to as dynamic power, results from the charging and discharging of the voltage saved in node capacitance of the circuit. We use Φ^S and Φ^D to denote static and dynamic power, respectively. In a router, NPUs operate in two possible states, namely "idle" and "busy". In the "idle" state, the power consumption is load-independent and equals to the static power, Φ^S. In the "busy" state, the power consumption is load-dependent and is composed of the static power Φ^S and dynamic power Φ^D. Consequently, the power consumed by a router can be expressed as follows:

$$\Phi = \begin{cases} \Phi^S, & \text{"idle" state} \\ \Phi^S + \Phi^D, & \text{"busy" state} \end{cases} \tag{3.18}$$

The dynamic power, Φ^D, can be further expressed as $\Phi^D = \gamma \cdot f^3$ [Gerards, 2014; Chen et al., 2012]. The parameter f denotes the clock frequency of the NPU processor and γ is a constant parameter, expressed in units of $Watts/GHz^3$.

3.7.1.1 Packet-based Energy Model For a given router, the dynamic power consumed by the data plane, Φ^D, is composed of two components, namely the *per-packet processing power* component, Φ_P, and the *per-byte store and forward power* component, $\Phi_{S\&F}$ [Vishwanath Member et al., 2014]. Both components are affected by the operational processor frequency, f. Φ_P represents the power consumed to process a given packet, regardless of the packet payload size. $\Phi_{S\&F}$, on the other hand, represents the power needed to receive, store, switch and transmit a packet. Contrary to Φ_P, which only depends on the number of instructions needed to process a packet (IPP_P), $\Phi_{S\&F}$ depends on the packet length, as packets with different lengths require different storage, switching time and transmission time, thereby consuming different amounts of power.

Let $IPB_{S\&F}$ denote the number of instructions required to process, store and forward a byte worth of data. Assuming a packet length of L *bytes*, the number of instructions required to process the packet is $IPP_{S\&F} = L \cdot IPB_{S\&F}$. Note that $IPB_{S\&F}$ is constant, as it only depends on the number of instructions to process a byte. Therefore, IPP_P can be expressed as a linear function of $IPB_{S\&F}$, namely $IPP_P = h \cdot IPB_{S\&F}$, where $h > 0$.

Let IPP represent the number of instructions to complete the processing, store, switch and transmission an entire packet with length L by a NPU at a given LC. We have $IPP = IPP_P + IPP_{S\&F} = (h+L) \cdot IPB_{S\&F}$. The NPU's processing, storage, switching and transmission time of a packet, $T_p = \frac{IPP}{IPS}$, where IPS represents the number of instructions executed by the NPU per second. IPS can be further expressed as $\frac{f}{CPI}$, where f denotes the operational frequency of the NPU and CPI represents the number of cycles per instruction. Therefore, $T_p = \frac{IPP \cdot CPI}{f} = \frac{\Theta \cdot (h+L)}{f}$, where $\Theta = CPI \cdot IPB_{S\&F}$.

Let $f_{j,i}$ denote the operational frequency of the active NPU j in LC i. In practice, the NPU only allows a number of manufacturer-specified discrete operational voltage levels, $V = \{V_1, ..., V_l, ..., V_M\}$. These discrete levels result in a corresponding set of discrete frequencies, $F = \{f_1, ..., f_l, ..., f_M\}$. Consequently, $f_{j,i}$ must be set to the smallest discrete frequency, $f_l (1 \leq l \leq M) | f_l \geq f_{j,i}$. The dynamic energy consumed by a successful packet

transmission with length L at NPU j in LC i is given by:

$$E^D_{j,i}(T_p) = \underbrace{\gamma_{j,i} \cdot f^3_{j,i}}_{\varphi^D_{j,i}} \cdot T_p = \gamma_{j,i} \cdot \Theta_{j,i} \cdot f^2_{j,i} \cdot (h_{j,i} + L) \tag{3.19}$$

3.7.1.2 Router-based Energy Model Assume that a router is equipped with Ψ LCs, each LC_i $(1 \leq i \leq \Psi)$ is equipped with n_i active NPUs. Let $T^B_{j,i} = \sum_{\forall} pT_p$ $(1 \leq j \leq n_i$ and $1 \leq i \leq \Psi)$ denote the busy time interval during which NPU j in LC i processes, stores, switches and transmits packets over the entire router's operation time, T, which includes idle and busy periods. The energy consumption of the router, over T, can be expressed as $E(T) = E^S(T) + E^D(T^B)$, where $E^S(T)$ represents the energy consumed due to static power during T and $E^D(T^B)$ represents the energy consumed due to dynamic power over the busy period, T^B. These energy components can be expressed as:

$$\begin{aligned} E^S(T) &= \Phi^S \cdot T \\ E^D(T^B) &= \sum_{i=1}^{\Psi} \sum_{j=1}^{n_i} E^D_{j,i}(T^B_{j,i}) \end{aligned} \tag{3.20}$$

$E^D_{j,i}$ represents the energy consumption by NPU j in LC i due to dynamic power, over the busy period, $T^B_{j,i}$. Note that $E^D_{j,i}$ depends on the dynamically changing frequencies used to process, store, switch and transmit a given packet, based on the scheduler's scaling decision.

Let $Z_{j,i}$ be the amount of time intervals at NPU j in LC i over time T, and $\tau_1, \cdots, \tau_k, \cdots, \tau_{Z_{j,i}}$, $1 \leq k \leq Z_{j,i}$, represent the frequency time slots at NPU j in LC i. Assuming $f_{j,i}(\tau_0)$ is the initial frequency. The frequency of NPU j at LC i, over the time interval τ_k, is $f_{j,i}(\tau_{k-1})$, where $1 \leq i \leq \Psi$, $1 \leq j \leq n_i$ and $1 \leq k \leq Z_{j,i}$. Let $D_{j,i}(\tau_k)$ denote the number of packets serviced by NPU j in LC i over the time interval τ_k. According to Eq. 3.19, the total dynamic energy consumed by $D_{j,i}(\tau_k)$ packets over the busy period $T^B_{j,i}(\tau_k) = \sum_{p \in D_{j,i}(\tau_k)} T_p$ during the time interval τ_k can be expressed as:

$$E^D_{j,i}(T^B_{j,i}(\tau_k)) \;=\; \gamma_{j,i} \cdot \Theta_{j,i} \cdot f^2_{j,i}(\tau_{k-1}) \cdot D_{j,i}(\tau_k) \cdot (h_{j,i} + \overline{L}_{j,i}(\tau_k)) \tag{3.21}$$

The parameter $\overline{L}_{j,i}(\tau_k)$ represents the average length of the packets serviced at NPU j

in LC i over the interval τ_k. The energy consumed by the router over the total operational period, $T = \sum_{k=1}^{Z_{j,i}} \tau_k$, is derived in Eq. 3.22.

$$E(T) = P^S \cdot T + \sum_{i=1}^{\Psi} \sum_{j=1}^{n_i} \sum_{k=1}^{Z_{j,i}} \gamma_{j,i} \cdot \Theta_{j,i} \cdot f_{j,i}^2(\tau_{k-1}) \cdot D_{j,i}(\tau_k) \cdot (h_{j,i} + \overline{L}_{j,i}(\tau_k)) \tag{3.22}$$

To validate the above router energy model, without loss of generality, we derive the total energy consumed by a router for two special cases: the first case assumes a set of Ψ homogeneous LCs, while the second assumes all LCs have the same number of active NPUs. Note that, in both cases, the packet energy consumption can be expressed as $E_p = EP_P + EB_{S\&F} \cdot \overline{L}$, where EP_P, expressed in $nJ/packet$, denotes the per-packet processing energy, and $EB_{S\&F}$, expressed in $nJ/byte$, denotes the per-byte store and forward energy, and $\overline{L}$ is the average packet length [Vishwanath Member et al., 2014].

Using the above model, we can derive an expression for h and γ, for a homogeneous network. Let $EB_{P_{max_{j,i}}} = EP_{P_{max}}$, $EB_{S\&F_{max_{j,i}}} = EB_{S\&F_{max}}$, $f_{max_{j,i}} = f_{max}$. Consequently, $h_{j,i} = h$, $\gamma_{j,i} = \gamma$, $\Theta_{j,i} = \Theta$, where $1 \leq j \leq n_i$, $1 \leq i \leq \Psi$. According to Eq. 3.19, we can compute $h = \frac{EP_{P_{max}}}{EB_{S\&F_{max}}}$ and $\gamma = \frac{EB_{S\&F_{max}}}{\Theta \cdot f_{max}^2}$. The router total energy for the simple homogeneous case is expressed in Eq. 3.23. The same method can be used to compute $\gamma_{j,i}$ and $h_{j,i}$, for a network of heterogeneous LCs.

$$E(T) = P^S \cdot T + \sum_{i=1}^{\Psi} \sum_{j=1}^{n_i} \sum_{k=1}^{Z_{j,i}} \frac{f_{j,i}^2(\tau_{k-1})}{f_{max}^2} \cdot D_{j,i}(\tau_k) \cdot (EP_{P_{max}} + EB_{S\&F_{max}} \cdot \overline{L}_{j,i}(\tau_k)) \tag{3.23}$$

Suppose that each LC has the same number of active NPUs. Furthermore, assume that all NPUs in LCs synchronously process their incoming traffic with average packet length $\overline{L}$, use the same speed-scaling strategy over the operational time interval, T. We set $Z_{j,i} = Z$, $n_i = n$, $\overline{L}_{j,i} = \overline{L}$ and $D_{j,i}(\tau_k) = D(\tau_k)$, where $1 \leq j \leq n$, $1 \leq i \leq \Psi$, $1 \leq k \leq Z$. The above energy model in Eq. 3.23 can be further simplified to:

$$E(T) = P^S \cdot T + \Psi \cdot n \cdot \frac{(EP_{P_{max}} + EB_{S\&F_{max}} \cdot \overline{L})}{f_{max}^2} \cdot \sum_{k=1}^{Z} f^2(\tau_{k-1}) \cdot D(\tau_k) \tag{3.24}$$

Eq. 3.23 and Eq. 3.24 demonstrate that adjusting the frequency, as opposed to using the maximum frequency, further reduces energy consumption. The following simulation study will be used to further determine the impact of dynamically adjusting frequencies on energy consumption.

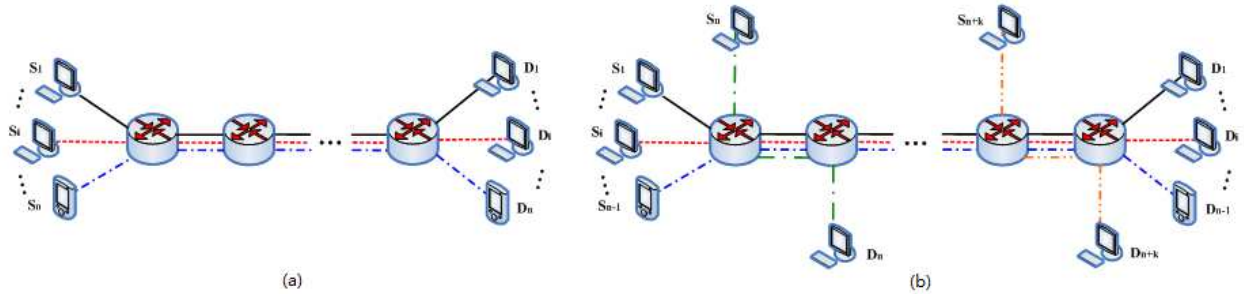

Figure 3.4: Two network topology models: (a) dumbbell, and (b) parking lot.

3.7.2 Simulation Setup

Simulation is an important tool in studying the performance of network protocols. The main objectives of this simulation-based performance analysis are threefold. The first objective aims to assess the sensitivity of each proposed scheme to its main parameters and how these parameters are correlated, particularly, its aggressivity to energy savings and network behavior predictability. The second objective is to carry out a comparative analysis of the proposed schemes, with respect to energy saving and adherence to QoS performance. The last objective is to compare the performance of the different schemes to similar schemes proposed in the literature.

The topology used in simulation-based performance analysis of network protocols often influences the outcome of the experiment. Consequently, the use of realistic network topologies to accurately capture the main behavior, dynamics and performance objectives is critical to producing realistic simulation results. Several studies were carried out to assess the viability of different types of topologies used in network simulation [Hayes et al., 2014; Jonckheere et al., 2002; Shah et al., 2003; Móczár et al., 2014]. Based on these studies, the *dumbbell* and *parking lot* topologies emerged as two promising models to capture the

behavior and performance of a large variety of applications, ranging from TCP applications over the Internet to multimedia applications, home networking, and transportation systems. A Dumbbell topology consists of a number of traffic hosts, attached to an inbound and an outbound switch. The two switches are connected by a **single** communication link. The parking lot topology is similar to the dumbbell topology, except that traffic hosts are also attached to intermediate routers between the inbound and outbound routers. In this study, we use an extended configuration of these topologies, by allowing multiple links between the inbound and outbound routers. Furthermore, the bandwidth and latency of the link are configurable. Fig.3.4 depicts the extended dumbbell and parking lot topologies used in this performance analysis study.

In Figure 3.4, S and D, denote the end-hosts, and the intermediate nodes between S and D are energy-saving routers. The capacities of links between all the routers are 10 $Gbps$. The routers implement FIFO scheduling and DropTail queuing. The propagation delays between the sources and the destinations are 40 ms, which is equivalent to the time the light travels from the east coast to west coast. In each router, all LCs are configured with multiple NPUs, each using a specific QoS-aware DVFS scheduler. In order to simulate real scenarios, Huawei CX600-X3 Metro Router model [Group, 2007], supporting $10GE$ LCs, is used. We further assume that each $10GE$ port provides 250 ms worth of traffic buffering. This results in processor buffers of approximately $250ms \times 10Gbps$, which is roughly 250000 packets, assuming the average packet size of 1250 bytes. The range of operating frequencies, $[1.6GHz,\ 2.4GHz]$, for a given NPU, is based on Intel XEON DPDK specification [Intel, 2012].

Table 3.2 describes the main simulation parameters used in this simulation study. According to [Simpson, 2006; Cha], Table 3.3 specifies three traffic source models, namely one constant bit rate (CBR) model: CBR Video, and two variable bit rate (VBR) models: VBR VoIP and VBR Data, which satisfy Pareto and Exponential On/Off distribution, respectively. Table 3.4 summarizes the different schedulers analyzed and compared in this simulation study. In addition to the QoS-aware schedulers, we also implemented a generic scheduler with no DVFS capabilities, called NoDVFS, as the experimental baseline, which uses the maximum frequency.

Table 3.2: Main simulation parameters and conditions.

Items	Simulation Parameters	Simulation Conditions
Router	Router Node	Metro Router [Vishwanath Member et al., 2014]
	NIC port	$10GE$
	Operating Frequency (GHz)	$1.6 \sim 2.4$
	$CPI(cycles/instruction)$	1.2
	$EP_{P_{max}}(nJ/pkt)$	1375 [Vishwanath Member et al., 2014]
	$EB_{S\&F_{max}}(nJ/byte)$	14.4 [Vishwanath Member et al., 2014]
	$P^S(Watts)$	352 [Vishwanath Member et al., 2014]
Packet	Packet Max size (bytes)	1500
	$IPB(instructions/byte)$	1.5
Queue	Service Discipline	FIFO
	Queuing Management Discipline	DropTail
Network	Topology Model	3-hop dumbbell
		4-hop parking lot
	Network Traffic *Load*	$0.5, \cdots, 0.9$
	Propagation Delay (ms)	dumbbell: 40; parking lot: 40
	Traffic Model	Video:VoIP:Data

Table 3.3: Traffic source models and specifications.

Flow Type	**Load Percentage**	T_{On} **(ms)**	T_{Off} **(ms)**	**Peak Rate**	β
Video	50%	NA	NA	$10Mbps$	NA
VoIP	20%	400	400	$64Kbps$	1.1
Data	30%	40	360	$256Kbps$	NA

Table 3.4: Speed scaling schedulers.

Scheme	**Metric**	**Scaling Factor**	**Scaling Strategy**
NoDVFS	None	None	$f(t) = f_{max}$
EWMAP [Tucker et al., 2009]	Predicted Load with $w_a = 0.2$	$\overline{\rho}$	$f(\tau_k) = max\left(f_{min}, min(f_{max} \cdot \overline{\rho}(\tau_k), f_{max})\right)$
LA	Average Load	ρ	$f(\tau_k) = max\left(f_{min}, min(f_{max} \cdot \rho(\tau_k), f_{max})\right)$
$\bar{\text{L}}$A	Predicted Load with $w_a(\tau_k)$	$\overline{\rho}$	$f(\tau_k) = max\left(f_{min}, min(f_{max} \cdot \overline{\rho}(\tau_k), f_{max})\right)$
sQLA	Current Queue Length	$(\frac{q+1}{Q+1})^{\eta}$	$f(\tau_k) = max\left(f_{min}, f_{max} \cdot (\frac{q(t_k)+1}{Q+1})^{\eta}\right)$
s$\bar{\text{Q}}$LA	Predicted Queue Length	$(\frac{\overline{q}+1}{Q+1})^{\eta}$	$f(\tau_k) = max\left(f_{min}, f_{max} \cdot (\frac{\overline{q}(\tau_k)+1}{Q+1})^{\eta}\right)$
mQLA	Current Queue Length	$(\frac{(q-ql)+1}{(qh-ql)+1})^{\eta}$	$f(\tau_k) = \begin{cases} f_{min}, & \text{if } q(t_k) \leq ql \\ f_{max}, & \text{if } q(t_k) > qh \\ f_{min} + (f_{max} - f_{min}) \cdot (\frac{(q(t_k) - ql) + 1}{(qh - ql) + 1})^{\eta}, & \\ & \text{if } ql < q(t_k) \leq qh \end{cases}$
m$\bar{\text{Q}}$LA	Predicted Queue Length	$(\frac{(\overline{q}-ql)+1}{(qh-ql)+1})^{\eta}$	$f(\tau_k) = \begin{cases} f_{min}, & \text{if } \overline{q}(\tau_k) \leq ql \\ f_{max}, & \text{if } \overline{q}(\tau_k) > qh \\ f_{min} + (f_{max} - f_{min}) \cdot (\frac{(\overline{q}(\tau_k) - ql) + 1}{(qh - ql) + 1})^{\eta}, & \\ & \text{if } ql < \overline{q}(\tau_k) \leq qh \end{cases}$
QLDA	Predicted Queue Length & Packet Delay (EWMA & GPR[3])	$(\frac{\overline{q}-ql}{qh-ql})^{\eta(\rho)}$	$f(\tau_k) = \begin{cases} f_{min}, & \text{if } \overline{q}(\tau_k) \leq q_l \\ f_{max}, & \text{if } \overline{q}(\tau_k) > q_h \\ f(\tau_{k-1}), & \\ & \text{if } q_l < \overline{q}(\tau_k) \leq q_h \text{ and } \left\lvert\overline{d}(\tau_k) - d^T\right\rvert \leq \tilde{dv}(\tau_k) \\ f_{min} + (f_{max} - f_{min}) \cdot (\frac{\overline{q}(\tau_k) - ql}{qh - ql})^{\eta(\rho(\tau_k))}, & \\ & \text{if } q_l < \overline{q}(\tau_k) \leq q_h \text{ and } \left\lvert\overline{d}(\tau_k) - d^T\right\rvert > \tilde{dv}(\tau_k) \end{cases}$

Table 3.5: Impact of η on ESP, AED, DJB and PLR of s$\bar{\text{Q}}$LA and m$\bar{\text{Q}}$LA ($q_l : q_h = 4\% : 80\%$) under traffic load $\rho = 0.9$ and $\tau = 1\ ms$.

	Dumbbell model						**Parking lot model**					
Scheme	**s$\bar{\text{Q}}$LA**			**m$\bar{\text{Q}}$LA**			**s$\bar{\text{Q}}$LA**			**m$\bar{\text{Q}}$LA**		
η	0.04	0.05	0.06	0.14	0.15	0.16	0.04	0.05	0.06	0.14	0.15	0.16
$ESP(\%)$	4.53	4.84	5.15	4.63	4.70	4.85	5.93	6.12	6.32	5.86	5.99	6.21
$AED(ms)$	103.68	137.09	168.73	126.6	134.4	153.15	96.77	126.11	156.69	123.38	129.39	150.42
$DJB(ms)$	2.32	7.09	21.39	4.74	9.76	11.48	1.08	7.84	10.78	5.36	8.71	10.55
$PLR(\%)$	0	0	0	0	0	0	0	0	0	0	0	0

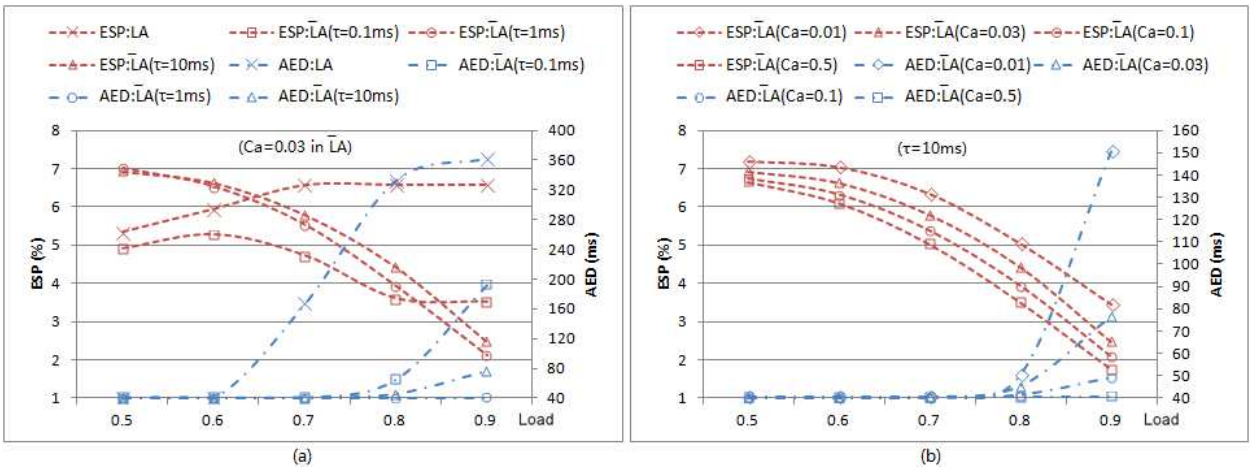

Figure 3.5: ESP and AED comparisons for (a) Load-aware schemes with different τ, and (b) $\bar{\text{L}}$A scheme with different c_a.

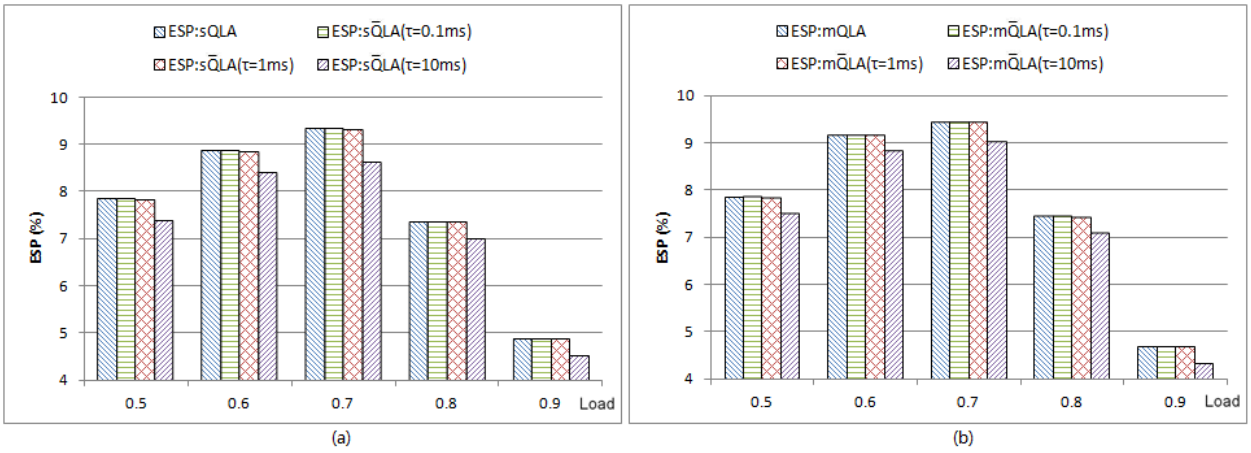

Figure 3.6: ESP comparisons for (a) sQLA/sQ̄LA schemes with $\eta = 0.05$, and (b) mQLA/mQ̄LA schemes with $\eta = 0.15$ under different τ.

The ITU G.114 specification recommends less than 150 ms one-way end-to-end delay for high-quality real-time traffic such as voice and video. In order to assure a good quality of the above traffic models, measures of the QoS parameters must respect the following values [Szigeti and Hattingh, 2005; ITU-T, 2003; Ellis et al., 2003]. In our simulation, we consider the following QoS requirements to evaluate energy saving percentage (ESP), average end-to-end delay (AED) for all discussed QoS-aware schemes.

- 150 ms as the average end-to-end delay threshold (AEDT) [ITU-T, 2003],
- 30 ms as the delay jitter bound (DJB),
- 1% as the packet loss rate (PLR) threshold.

3.7.3 Sensitivity to the main parameters of Load-aware Schemes

In this section, we carried a series of experiments to explore the sensitivity of Load-aware schemes to different parameters. The objective is to fine tune the main parameters of these schemes to achieve the balance between the high energy saving and the QoS performance.

3.7.3.1 Sensitivity to τ Different values of the monitoring period τ are tested to determine the sensitivities of the Load-aware schedulers to DVFS adjustment. Choosing a small prediction period, τ, in Load-aware schemes could suffer the overhead impact of the back-to-back undesirable DVFS adjustment. In this experiment, we study the impact of the prediction period τ on the energy saving and the packet average delay of the $\bar{\text{L}}$A scheme in the range $[0.1, 10]$ ms. Different from the $\bar{\text{Q}}$LA schemes, the $\bar{\text{L}}$A scheme exhibits sensitivity to τ. Fig.3.5 (a) shows that the $\bar{\text{L}}$A scheme can achieve adequate energy saving without QoS violence when τ is set as 10 ms.

3.7.3.2 Sensitivity to c_a In the $\bar{\text{L}}$A scheme, the EWMA algorithm uses a prediction factor c_a to dynamically adjust smooth filter w_a to predict the traffic load. Different values of c_a in the range $[0.01, 0.50]$ are tested. Fig.3.5 (b) depicts that the energy saving and average packet end-to-end delay are very sensitive to c_a. The energy saving increases with the value of c_a decreases. However, a small value of c_a could lead to a sharp delay increase and a high packet loss rate, such as the scenario with $c_a = 0.01$ in $\bar{\text{L}}$A scheme. In order to guarantee the QoS requirements, $c_a = 0.03$ is selected under $\tau = 10$ ms for the following analysis.

3.7.4 Sensitivity to the main parameters of QL-aware Schemes

Similarly, we carried a series of experiments to explore the sensitivity of QL-aware, single- and multi-thresholds schemes to different parameters.

3.7.4.1 Sensitivity to η The first experiment is designed to study the schedulers' sensitivity to the aggressivity factor, η. A series of values in the range $[0.01, 0.20]$ under the high traffic load, $\rho = 0.9$, is tested for QL-aware schemes. The results show that the energy saving and the average packet delay both increase when the value of η increases, a larger value of η can save more energy, it, however, could lead to a dramatic delay increase, especially under the high traffic load, as displayed in Table 3.5. Therefore, given a NPU's frequency range and a network model, the upper bound value of η can be found to achieve adequate energy saving without QoS violence. Setting $\eta = 0.05$ for sQLA/s$\bar{\text{Q}}$LA, and $\eta = 0.15$ for mQLA/m$\bar{\text{Q}}$LA, we can get the largest energy saving under the acceptable QoS requirements in the respective family schemes.

In this chapter, we mainly analysis 3-hop dumbbell and 4-hop parking lot models. In the real world, however, there are more realistic and complex topologies. Therefore, given the QoS requirements, the network model, the routing path, the number of hops along the routing path and the application traffic load could be important impact factors to the upper bound of η. Our future work will further explore these issues.

3.7.4.2 Sensitivity to τ The second experiment is designed to study the schedulers' sensitivity to the rate of DVFS adjusting. Assuming a frequency range, $[f_{min}, f_{max}]$, a small frequency adjustment interval creates more opportunities for a more accurate adjustment of the frequency, based on the current or average queue length. A small interval, however, increases the frequency adjustment overhead. A large frequency adjustment interval reduces the overhead required to adjust frequencies, but fails to capture more accurately the current level of congestion. Fig.3.6 (a,b) depicts the energy saving percentage for the different QL-aware schemes under the acceptable average end-to-end packet delay, using different frequency adjustment interval, τ. The results show that two $\bar{\text{Q}}$LA schemes with the respective aggressivity factors are not sensitive to τ when the value of τ is under 1 ms. Therefore, $\tau = 1$ ms is selected for the rest of the experiments.

3.7.4.3 Sensitivity to c_q In sQ̄LA and mQ̄LA schemes, the EWMA algorithm uses a constant parameter, c_q, to adaptively adjust the smooth filter, w_q. Different values of c_q in the range $[0.01, 0.50]$ are tested in both sQ̄LA scheme and mQ̄LA scheme. The results show that the energy saving and packet average end-to-end delay are not sensitive to c_q. When the value of c_q increases, the energy saving increases slightly. Therefore, the value of c_q is set to 0.5 in our simulation study.

3.7.4.4 Sensitivity to q_l and q_h In this experiment, the value of η is set to 0.15, and the value of τ is set to 1 ms, while the thresholds, q_l and q_h are varied, as described in Table 3.6. Four combinations of q_l and q_h are tested to study the impact of the queue-length thresholds on energy saving and average packet delay in the mQ̄LA scheme. As shown in Table 3.7, although the energy saving is not very sensitive to q_h, a higher q_h can save more energy. On the other hand, the packet delay is very sensitive to q_l, it increases dramatically with the value of q_l increases. Therefore, adjusting queue-length thresholds, i.e. q_l and q_h, can optimize the effectiveness and efficiency of the mQ̄LA scheme. Our experiment shows that setting $q_l : q_h = 4\% : 80\%$ provides an adequate balance between the high energy saving and the acceptable QoS requirements.

Table 3.6: Four combinations of (q_l, q_h) in the mQ̄LA scheme.

	$q_h = 60\% \cdot Q$	$q_h = 80\% \cdot Q$
$q_l = 4\% \cdot Q$	(1.0 E+4, 1.5 E+5)	(1.0 E+4, 2.0 E+5)
$q_l = 10\% \cdot Q$	(2.5 E+4, 1.5 E+5)	(2.5 E+4, 2.0 E+5)

Table 3.7: Impact of $q_l : q_h$ on ESP and AED in mQ̄LA scheme.

	ESP (%)				**AED (ms)**			
$q_l : q_h$	4% : 60%	4% : 80%	10% : 60%	10% : 80%	4% : 60%	4% : 80%	10% : 60%	10% : 80%
$\rho = 0.7$	9.43	9.45	9.67	9.68	79.72	79.72	126.77	126.77
$\rho = 0.8$	7.43	7.44	7.69	7.69	78.49	79.05	130.14	130.66
$\rho = 0.9$	4.56	4.70	4.80	4.94	120.63	134.40	162.82	176.76

3.7.5 Comparative analysis

The following mainly analyzes and compares the performance of the two basic family scheduling schemes, namely Load-aware family and QL-aware family, in terms of energy gain such as energy saving percentage (EAP) and average end-to-end delay (AED).

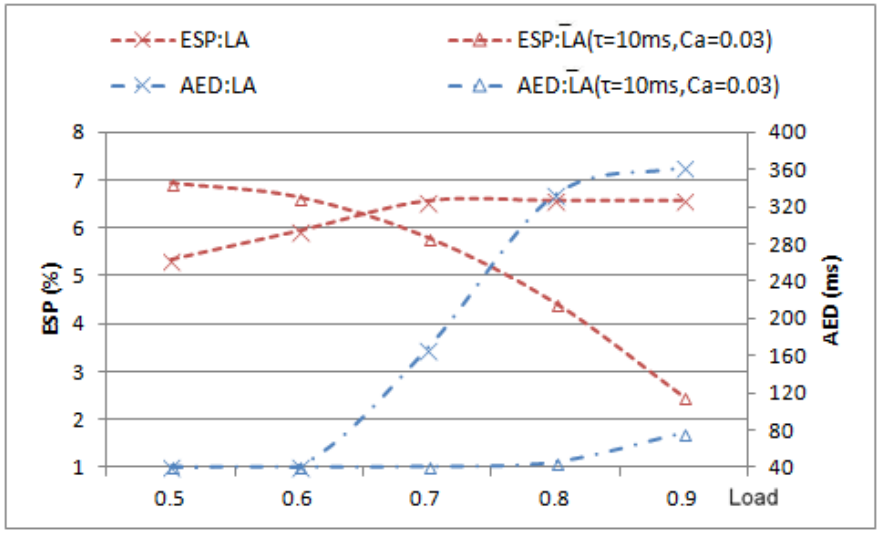

Figure 3.7: ESP and AED comparisons for two Load-aware schemes.

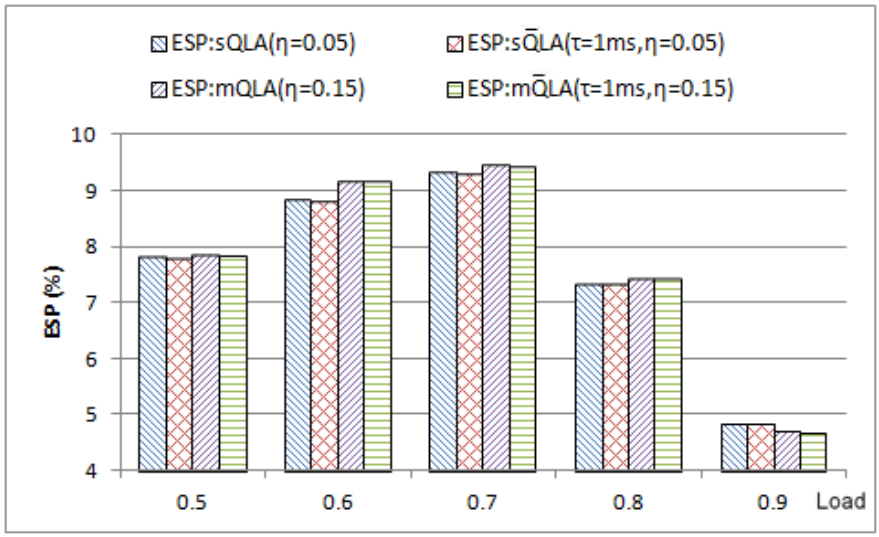

Figure 3.8: ESP comparison for four QL-aware schemes.

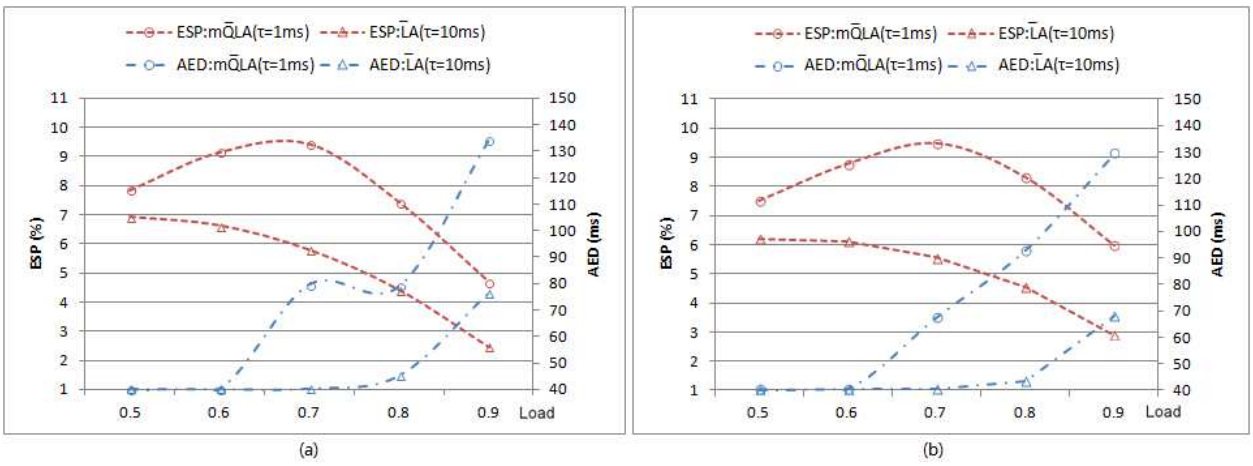

Figure 3.9: ESP and AED comparisons between mQ̄LA ($\eta = 0.15$, $q_l : q_h = 4\% : 80\%$) and L̄A ($c_a = 0.03$) for (a) dumbbell model, (b) parking lot model.

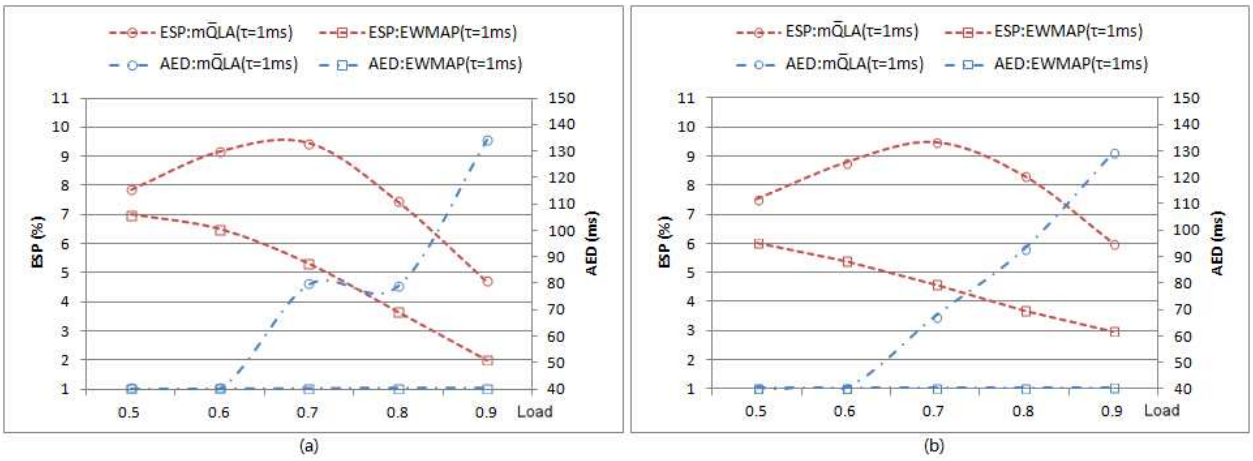

Figure 3.10: ESP and AED comparisons between mQ̄LA ($\eta = 0.15$, $q_l : q_h = 4\% : 80\%$) and EWMAP ($w_a = 0.2$) for (a) dumbbell model, (b) parking lot model.

3.7.5.1 The class of Load-aware schemes As illustrated in Figure 3.7, the LA scheme has a severe QoS degradation, leading to heavy packet dropping and huge packet delay missing. However, the traffic load prediction based on the EWMA algorithm is useful in controlling the network delay and improving network performance. The results show that properly setting the values of c_a and τ, such as $c_a = 0.03$ and $\tau = 10ms$, the L̄A scheme can achieve a better balance between the high energy saving and the QoS performance compared

to the LA scheme. When $\rho = 0.5$, $\bar{\text{L}}$A can save up to 7.0% energy with AED of 40.25 ms and DJB of 0.75 ms, when $\rho = 0.9$, it can save around 2.5% energy with AED of 76.43 ms and DJB of 4.95 ms. The parking lot model shows results with the same trend and range in Figure 3.9.

3.7.5.2 The class of QL-aware schemes As shown in Figure 3.8, among QL-aware schedulers, the energy saving of s$\bar{\text{Q}}$LA and m$\bar{\text{Q}}$LA under $\tau = 1$ ms are very approximate to sQLA and mQLA with their respective aggressivity factor. Although sQLA and s$\bar{\text{Q}}$LA has very slight higher energy saving than mQLA and m$\bar{\text{Q}}$LA at the high traffic load, $\rho = 0.9$, mQLA and m$\bar{\text{Q}}$LA with $\eta = 0.15$ and $q_l : qh = 4\% : 80\%$ are potential to save more energy than sQLA and s$\bar{\text{Q}}$LA with $\eta = 0.05$ under the QoS requirements in general. Furthermore, m$\bar{\text{Q}}$LA can achieve almost the same energy saving as mQLA, with lower DVFS switching overhead. Fig.3.8 shows that m$\bar{\text{Q}}$LA saves more than 4% energy compared to the NoDVFS scheme in dumbbell network model. For $\rho = 0.7$, m$\bar{\text{Q}}$LA can save up to 9.5% energy with AED of 79.72 ms and DJB of 1.81 ms. Even although $\rho = 0.9$ results in 134.40 ms AED and 9.76 ms DJB, the corresponding energy-saving percentage in m$\bar{\text{Q}}$LA is up to 4.7%. For parking lot model, we have results with the same trend and range, as displayed in Figure 3.9.

3.7.5.3 Cross class comparative analysis As discussed above, the two most effective and efficient schemes from the QL-aware and Load-aware DVFS classes are the potential to save significant energy without QoS violence given the appropriate parameters. Fig.3.9 (a) and (b) depict that the m$\bar{\text{Q}}$LA scheme and the $\bar{\text{L}}$A scheme in two different network topologies, i.e. dumbbell and parking lot, have the same trend in the energy saving and the packet average delay, whereby the m$\bar{\text{Q}}$LA scheme with $\eta = 0.15$, $q_l : q_h = 4\% : 80\%$ and $c_q = 0.5$ under $\tau = 1$ ms can provide up to around 9.5% energy saving, and the m$\bar{\text{Q}}$LA scheme can achieve up to 4% more energy saving than the $\bar{\text{L}}$A scheme without QoS violence.

In general, it is hard for Load-ware schemes to control the QoS performance accurately, especially in the high traffic load when Load-ware schemes are easy to violate QoS requirements. On the contrary, the QL-aware scheduler, m$\bar{\text{Q}}$LA, can more accurately control the QoS performance through adjusting queue length thresholds. Therefore, QL-aware schemes

have an advantage over Load-aware schemes in balancing high energy saving and QoS requirements.

3.7.5.4 Comparison with the related work In [Tucker et al., 2009], Mandviwalla and Tzeng propose three Load-aware predictors to reduce energy consumption in LCs, in which the most effective Load-aware predictor is called EWMAP. Different from our proposed $\bar{\text{L}}$A scheme, EWMAP uses EWMA algorithm with a fixed load smooth filter, w_a (i.e. μ in [Tucker et al., 2009]), to predict link utilization over a constant perdition interval, τ (i.e. PI in [Tucker et al., 2009]), to control the execution rates of LCs, aiming to achieve energy saving. According to [Tucker et al., 2009], a fixed load prediction factor $w_a = 0.2$ is recommended in the EWMAP scheme. Through testing different values of w_a in the range $[0.01, 0.50]$, $w_a = 0.2$ is verified to be the best choice to achieve the high energy saving without QoS violence in the EWMAP scheme. In addition, different values of the prediction period, τ, in the range $[0.1, 10]$ ms are also tested to determine the sensitivities of the EWMAP scheduler to DVFS adjustment. Different from m$\bar{\text{Q}}$LA, the Load-based EWMAP scheme exhibits sensitivity to τ. The results show that the EWMAP scheme can achieve the largest energy saving without QoS violence when τ is set to be 1 ms.

Using the same router-based energy model, we compare the m$\bar{\text{Q}}$LA scheme with the EWMAP scheme under $\tau = 1$ ms in different network topologies, as shown in Figure 3.10 (a) and (b). The results show that the EWMAP scheme can save router energy from 2% to 7% according to different traffic load, however, the m$\bar{\text{Q}}$LA scheme can achieve up to 5% more energy saving than the EWMAP scheme without QoS violence. Therefore, the m$\bar{\text{Q}}$LA scheme outperforms the other Load-aware schemes in achieving significant energy saving under the acceptable QoS requirements.

3.7.6 Sensitivity to the main parameters of QLDA

Furthermore, we carried out a series of experiments to do the sensitivity analysis of the proposed QLDA scheme to different parameters.

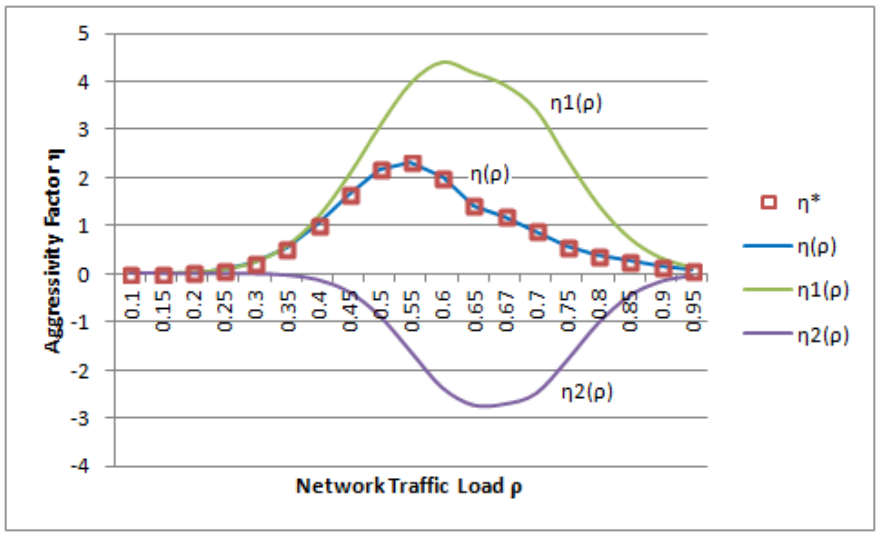

Figure 3.11: The aggressivity factor $\eta(\rho)$.

3.7.6.1 Sensitivity to η The first experiment is designed to study the sensitivity of the scheduler to the aggressivity factor η. The results show that the value of the aggressivity factor to achieve the highest energy saving depends on the network load. In order to determine the "optimum" η^*, a series of simulation experiments where carried out, whereby for a given network load, ρ, multiple values of η are tested and the value which produces the highest energy saving, without violating the traffic QoS requirements, is selected. The experiments used two different network models, namely dumbbell and parking lot, assuming $f_{min} = 1.6\ GHz$ and $f_{max} = 2.4\ GHz$. Using Matlab, the two independent variables, η and ρ, are fitted by a Gaussian process regression (GPR) function of successive approximations, as illustrated Eq. 3.25. In this equation, (a_1, b_1, c_1) and (a_2, b_2, c_2) are GPR model parameters, where $(a_1, b_1, c_1) = (4.4, 0.6085, 0.1805)$ and $(a_2, b_2, c_2) = (-2.745, 0.6554, 0.1466)$. The fitted curve is depicted in Figure 3.11.

$$\begin{aligned} \eta(\rho) =& \eta_1(\rho) + \eta_2(\rho) \\ =& a_1 \cdot e^{-(\frac{\rho - b_1}{c_1})^2} + a_2 \cdot e^{-(\frac{\rho - b_2}{c_2})^2} \end{aligned} \tag{3.25}$$

Using the load-dependent values of η, generated by the GPR function, the two network topology models, dumbbell and parking lot, were used to assess the performance of QLDA

and determine the levels of energy saving it achieves, under different network loads, while maintaining acceptable QoS requirements. The results of these experiments are shown in Table 3.8.

Table 3.8: Impact of η on ESP, AED, DJB and PDMRT of QLDA scheme with $q_l : q_h = 4\% : 80\%$.

	Dumbbell model			**Parking lot model**		
Load ρ	0.7	0.8	0.9	0.7	0.8	0.9
$ESP(\%)$	9.82	7.83	4.76	9.76	8.68	6.16
$AED(ms)$	122.30	133.89	132.34	116.89	131.18	125.77
$DJB(ms)$	1.73	3.40	6.12	1.24	2.78	5.74
$PLR(\%)$	0	0	0	0	0	0

Table 3.9: Impact of $q_l : q_h$ on ESP and AED in the QLDA scheme under dumbbell model.

	ESP (%)				**AED (ms)**			
$q_l : q_h$	4% : 60%	4% : 80%	10% : 60%	10% : 80%	4% : 60%	4% : 80%	10% : 60%	10% : 80%
$\rho = 0.7$	9.75	9.82	9.77	9.85	111.19	122.23	143.33	150.77
$\rho = 0.8$	7.73	7.83	7.75	7.85	117.90	133.89	162.18	177.65
$\rho = 0.9$	4.60	4.76	4.63	4.76	117.03	132.34	158.62	174.17

3.7.6.2 Sensitivity to τ The second experiment is designed to study the scheduler's sensitivity to the rate of DVFS adjusting. The values of η for different traffic loads refer to Table 3.8. Under a frequency range, $[f_{min}, f_{max}]$, a small frequency adjustment interval creates more opportunities for a more accurate adjustment of the frequency, based on the queue length. A small interval, however, increases the frequency adjustment overhead. A large frequency adjustment interval reduces the overhead required to adjust frequencies, but fails to capture more accurately the current level of congestion. Fig.3.12 (a) and (b) depict the energy-saving percentage for the different network models, using different frequency adjustment interval, τ, under the range of $[0.01, 100]$ ms. The results show that QLDA, assuming $q_l = 4\% \times Q$ and $q_h = 80\% \times Q$, is not sensitive to τ when the value of τ is under 1 ms. Therefore, $\tau = 1$ ms is selected for the rest of the experiments.

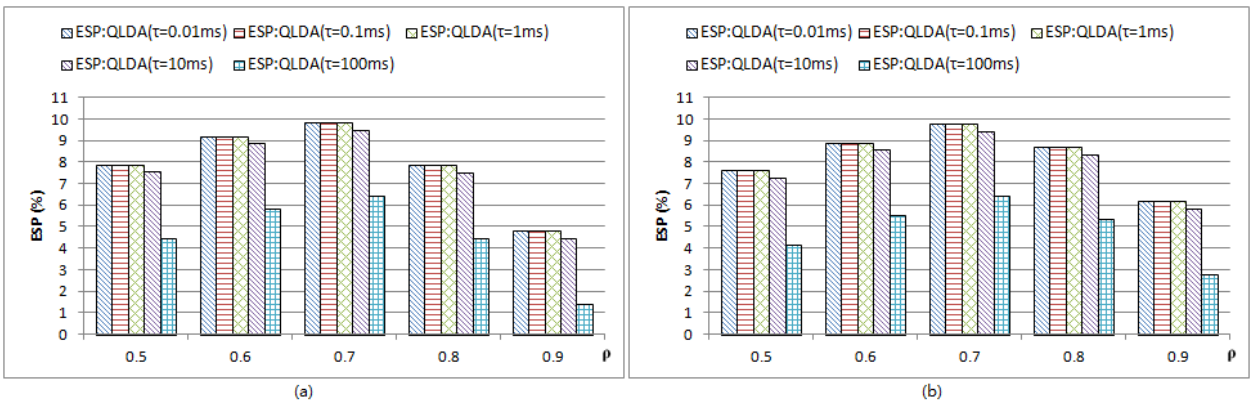

Figure 3.12: ESP comparisons for QLDA with different τ in (a) dumbbell model, and (b) parking lot model.

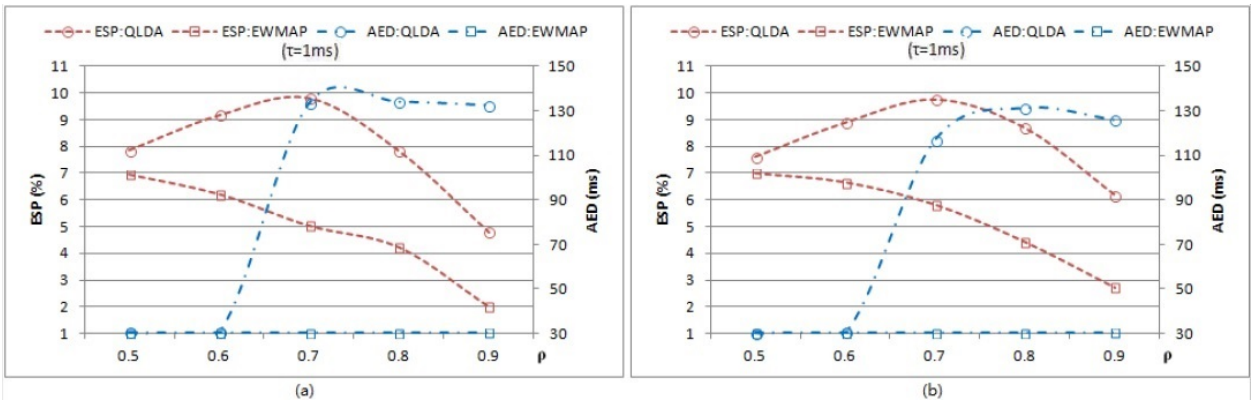

Figure 3.13: ESP and AED comparisons between QLDA ($q_l : q_h = 4\% : 80\%$) and EWMAP ($\mu = 0.2$) for (a) dumbbell model, (b) parking lot model.

3.7.6.3 Sensitivity to c_q QLDA scheme uses EWMA based algorithm with weight, w^q, to predict the queue length. The constant parameter c_q is used in the error prediction function to adaptively adjust w^q. Different values of c_q in the range $[0.01, 0.50]$ are tested in the QLDA scheme. The results show that the energy saving and the average end-to-end packet delay are not sensitive to c_q. When the value of c_q increases, the energy saving increases slightly. Therefore, $c_q = 0.5$ is selected for the following analysis.

3.7.6.4 Sensitivity to c_d Similarly, QLDA uses the EWMA algorithm to predict the average packet delay. The constant parameter c_d is used in the error prediction function to dynamically adjust the smooth filter w^d. Different values of c_d in the range $[0.01, 0.5]$ are tested in the QLDA scheme. The results show that the energy saving and the average end-to-end packet delay are not sensitive to c_d. When the value of c_d decreases, the energy saving increases slightly. Therefore, $c_d = 0.01$ is considered.

3.7.6.5 Sensitivity to q_l and q_h In this experiment, the value setting of η for different traffic load is based on the above GPR function, as shown in Figure 3.11, and the value of τ is set to 1 ms, while the thresholds, q_l and q_h, are varied, as described in Table 3.6. Four combinations of q_l and q_h are tested to study the impact of the queue length thresholds on energy saving and average packet delay. The results, shown in Table 3.9, indicate that, although the energy saving is not highly sensitive to q_h, a higher value of q_h leads to higher energy saving. The results also show that packet delay is highly sensitive to q_l, as the delay increases dramatically when the value of q_l increases. Therefore, adjusting queue-length thresholds can improve the effectiveness and efficiency of the QLDA scheme. The results show that for a dumbbell topology, the ratio $q_l : q_h = 4\% : 80\%$ leads to the highest energy savings, without violating QoS requirements. A similar outcome can be observed in the case of a parking lot model.

3.7.7 Comparative analysis

As mentioned above, the most effective Load-aware predictor, EWMAP, proposed in [Tucker et al., 2009], uses EWMA algorithm with a fixed load smooth filter, μ ($\mu = 0.2$ is recommended), to predict traffic load over a constant perdition interval, τ (i.e. PI in [Tucker et al., 2009]), to control the execution rates of LCs, thereby achieving energy saving. Different values of the prediction period, τ, in the same range $[0.01, 100]$ ms as the QLDA scheme are tested to determine sensitivities of the same EWMAP scheduler to DVFS adjustment. Different from QLDA, the Load-aware EWMAP scheme exhibits sensitivity to τ. The same results as above show that the EWMAP scheme achieves the maximal energy saving without

QoS violence when set τ be equal to 1 ms.

Using the same router-based energy model, we compare the proposed QLDA scheme with the EWMAP scheme under $\tau = 1$ ms in two different network models, as shown in Figure 3.13 (a) and (b). We found that two network models have the same trend in the energy saving and the average end-to-end packet delay in both QoS-aware schemes respectively. The results show that these two QoS-aware schemes are the potential to save significant energy. Under the same QoS requirements, the QLDA scheme with $q_l : q_h = 4\% : 80\%$ can provide up to 9.82% energy saving with AED of 122.30 ms and DJB of 1.73 ms, and 9.76% energy saving with AED of 116.89 ms and DJB of 1.24 ms, in the dumbbell model and the parking lot model, respectively. Although the EWMAP scheme leads to an increase in energy saving, from 2% to 7%, under different traffic loads, the results show that QLDA achieves up to 5% increase in energy saving than EWMAP, without violating QoS requirements. In addition, choosing a small prediction period in the Load-aware schemes could suffer the overhead impact of the back-to-back undesirable DVFS adjustment, as discussed above. However, under the same prediction period condition, since the scaling decisions depends on not only a given prediction period but also the predicted queue length and packet delay, QLDA shows much lower overhead compared with EWMAP scheme.

As an extended scheme of m$\bar{\text{Q}}$LA [Yu et al., 2015a], QLDA [Yu et al., 2015b] scheme adopts multiple queue length thresholds to accurately capture network congestion. In response to different levels of network congestion, different NPU-rate scaling strategies are used to determine when and how NPU execution rates are adjusted based on the predicted queue length and the estimated delay variance, aiming to achieve adequate energy savings under different traffic loads, without degrading delay performance. Besides, using the Gaussian regression model to generate the aggressivity factor of the scheduling strategy based on the traffic load is another improvement of m$\bar{\text{Q}}$LA. Therefore, QLDA displays more efficient and sufficient on balancing high energy saving and QoS requirements compared with m$\bar{\text{Q}}$LA, as depicted in Figure 3.10 and Figure 3.13 (note: μ in Figure 3.13 is w_a in Figure 3.10). And the related results show that QLDA can achieve energy saving of around 10%, as shown in Figure 3.13 (a), higher than m$\bar{\text{Q}}$LA scheme. And QLDA also outperforms m$\bar{\text{Q}}$LA in decreasing the overhead because of its more demanding conditions.

3.8 Conclusions

In this chapter, we proposed three families of QoS-aware DVFS-based schedulers and derived variants for each family, based on queue length and link utilization. The variants, in each family, differ in when and how decisions are used to adjust the execution rates. The objective is to fine tune the main parameters of these schemes to achieve the balance between the energy minimization and the QoS requirements. A thorough analysis of the proposed schemes, using NS2, has been carried out for different network environments and traffic loads. The simulation results show that all QoS-aware DVFS-based schemes have the potential for significant energy saving in high-speed networks with acceptable delay performance, and that QL-aware family schemes achieve, on average, higher energy savings than Load-aware family schemes. The m$\bar{\text{Q}}$LA scheme achieves the best results within these two basic families, with performance gains of up to 9.5% energy saving, while meeting the QoS performance of the supported applications. Furthermore, as the extension of m$\bar{\text{Q}}$LA, QLDA has an advantage over m$\bar{\text{Q}}$LA, which can achieve up to 10% energy saving while keeping the desired QoS performance. In general, it is hard for Load-aware schemes to accurately control the QoS performance, especially in the high traffic load. On the contrary, the QL-aware schedulers and the QL-based, Delay-aware scheduler, QLDA, can more accurately control the QoS performance through adjusting queue length thresholds. Specifically, QLDA, as an improvement of QL-aware family schemes, more efficiently balances high energy saving and QoS requirements and less overhead compared with other QL-aware schemes.

Compared to the existing speed scaling solutions, our proposed QoS-aware packet schedulers address the energy-performance challenge. By doing so, they not only save significant dynamic energy, but they also seek a balanced tradeoff between high network energy savings and acceptable levels of network QoS performance under different traffic loads, based on a comprehensive router-based energy model. In this chapter, we only analyzed two simple dumbbell and parking lot network topology models. In the real world, however, there are more realistic and complex network topologies. Therefore, given the QoS requirements, the network model, the routing path, and the character of traffic load could become the critical factors that would impact the upper bound of the corresponding aggressivity parameter in

a scheduling strategy design. They should be considered in future research work. Moreover, this chapter mainly discussed DVFS-based experimental power management, but is it possible to minimize energy consumption under the QoS constraints by a mathematical programming model theoretically? The next chapter will further explore this issue: DVFS-based algorithmic power management and its optimal energy strategies.

4.0 DVFS-based Power Management and Delay-aware, Optimal Energy Strategies

The ability to efficiently manage network resources to minimize energy consumption is critical to effectively address network congestion and end-to-end QoS guarantees. The previous chapter discusses DVFS-based experimental power management, which is used to sufficiently reduce energy consumption while meeting QoS requirements through applying different energy- and QoS-aware packet scheduling strategies. However, among a large number of dynamic power management solutions, a less expensive one is to use mathematical optimization models to formulate the energy saving problem [Orgerie et al., 2014]. For DVFS-enabled network components, the key issue is how to let their energy optimization problem subjects to the conflicting dual objectives of speed scaling and QoS requirements. In this chapter, we study DVFS-based algorithmic power management to explore this question. Given a traffic flow, each router along the routing path the flow is traveling must determine the node execution speed, at which it must process traffic to minimize energy consumption under the corresponding QoS constraints. By applying an existing scheduling policy, we propose a DVFS-based, delay-aware energy optimal strategy and its two heuristics to optimize the energy consumed by network components, which take into consideration the workload and the delay requirements across the routing path. The results show that the proposed strategy achieves up to 83.33% dynamic energy saving of total dynamic energy consumption and up to 26.76% power saving of total power consumption under QoS constraints.

4.1 Introduction

Minimizing power and energy consumption has become a critical objective in the design of future networks [Lange et al., 2009]. In 2009, the backbone energy consumption accounted for less than 10% of the overall network energy consumption, but this percentage is expected to increase to 40% in 2017 [Lange et al., 2009], and reach up to or even exceed 50% in 2020,

and thus will become unsustainable [Hinton et al., 2011]. Recent advances in networking and communications technologies paved the way for a new generation of faster and more powerful routers and switches, ushering in the proliferation of delay-bound IP applications. The need to support the quality-of-service (QoS) requirements of these emerging applications further compound the power and energy consumption problem, calling for new energy- and delay-aware approaches to traffic management and congestion control in future differentiated-service networks [Chabarek et al., 2008; Bolla et al., 2011b; Pierson, 2015; Bianzino et al., 2012; Zeadally et al., 2012].

A number of approaches have been proposed to reduce the energy consumed by network processing routers and interfaces [Pierson, 2015]. These approaches mostly aim at managing network resources, in response to traffic load, to minimize network energy consumption. However, the problem of minimizing energy consumption, while meeting the QoS-requirements of delay-sensitive applications, has received minimal attention [Addis et al., 2016]. To address this shortcoming, we propose an energy- and delay-aware traffic control and management framework and explore the design and performance assessment of a DVFS-enabled, energy- and delay-aware flow scheduling strategy to support delay-sensitive applications. More specifically, given a flow specification, which characterizes the flow's traffic rate and QoS performance requirements, a per-router *feasible* minimum and maximum delay values are computed. Using these values, the energy- and delay-aware problem is modeled as a *routing path energy-minimization problem* to determine, for each router along the path, the processing router execution speed and the flow delay that minimize energy without violating the flow's *end-to-end* delay requirement. The main contributions in the chapter are: (i) develop a model to compute a path-based energy consumption, taking into consideration both the static and dynamic energy components; (ii) build up a methodology to compute feasible lower and upper bound delays of a flow, based on the router's current traffic load; and (iii) propose a strategy to compute feasible per-router delays that meet the end-to-end delay requirements and minimize energy across the path. A simulation framework is used to assess the performance of the proposed strategy algorithm and its two heuristics in terms of two energy-efficient metrics, namely dynamic energy gain (DEG) and power gain (PG).

The rest of this chapter is organized as follows: the related work is reviewed in Sec-

tion 4.2. An existing scheduling policy is introduced in Section 4.3. The network and flow specifications are introduced in Section 4.4. The formulation of the energy- and delay-aware traffic control and management framework is discussed in Section 4.5. Within this framework, a methodology used for computing per-router delays is described. A path-based energy consumption model is then built up and a DVFS-enabled, energy- and delay-aware flow scheduling strategy and its two heuristics to compute the execution speed of a router and a per-router delay budget are proposed. The performance of the proposed strategy is assessed in Section 4.6. Finally, Section 4.7 presents the conclusion of this chapter.

4.2 Related Work

Energy-efficient scheduling techniques to reduce energy consumption can be broadly classified into two categories, namely *Sleep Mode* (SM) and *Dynamic Voltage and Frequency Scaling* (DVFS) [Pierson, 2015; Bolla et al., 2011b; Bianzino et al., 2012; Zeadally et al., 2012; Nedevschi et al., 2008]. Sleep Mode [Sabhanatarajan et al., 2008; Ghazisaeedi et al., 2012; Ghazisaeedi and Huang, 2015]. To reduce energy consumption, SM-based approaches selectively put system components into a power-efficient mode, by turning them off whenever they become idle. The shortcomings of the Sleep Mode based techniques is that once a computing or communication component is moved into a power-efficient mode, bringing the component back to an active or running mode incurs additional energy and latency, which may have a significant impact on the performance of the system. DVFS-based approaches dynamically adjust the processor's voltage and frequency, in response to workload variation, in order to minimize energy consumption [Mandviwalla and Tzeng, 2006]. Yu et al. [Yu et al., 2015a,b] propose three families of delay-aware packet scheduling schemes to dynamically control line cards' execution rates and reduce routers' energy consumption. Two congestion control metrics, namely *queue length* and *link utilization*, are used to assess the level of network congestion and adjust the processor's frequency to sufficiently reduce energy consumption. The results show that queue length(QL)-aware schemes outperform Load-aware schemes and achieves higher energy savings. It has been found that the main

challenge of DVFS-based power management techniques stems from the difficulty of determining the minimum voltage to meet a particular component performance level. As such, most of the proposed experimental power management schemes do not guarantee the end-to-end delay requirements of the underlying application to achieve maximal energy saving. To this end, the demand for the less expensive DVFS-based algorithmic power management strategies through optimization models increases.

In Gupta et al. propose sleep modes to reduce network energy consumption [Gupta et al., 2004; Gupta and Singh, 2007b]. The focus of this work, however, is on Local Area Networks (LANs). This limits significantly the applicability of the proposed approach to backbone networks, where inter-packet time is too short to warrant putting links into sleep mode. In [Nedevschi et al., 2008], Nedevschi et al. propose a method to shape traffic into bursts and create link sleeping opportunities between bursts. It also uses link rate adaptation to traffic load to save network energy. The effectiveness of the scheme depends on the inter-packet arrival time and the burst size. In [Chabarek et al., 2008], Chabarek et al. explore power-awareness in the design of networks and routing protocols. The proposed approach, however, does not lead to a specific power-aware routing design. In [Heller et al., 2010], Heller et al. propose ElasticTree, to optimize the energy consumption of a data center network by turning off unnecessary links and switches during off-peak hours. The solution is specific to the class of data center tree-based topologies. In [Zhang et al., 2010a], Zhang et al. propose the GreenTE framework for a network-level power management approach to optimize the number of links that can be put into sleep mode in order to maximize network energy saving. This scheme, however, is prone to large delay variation, which may lead to unacceptable delay-jitter.

The discussed framework in this chapter addresses these shortcomings and proposes several DVFS-based, delay-aware optimal energy strategies to minimize energy consumption while adhering to the end-to-end delay requirements of the underlying traffic flows, within this framework. All of these strategies follow a given scheduling policy, which is introduced in the coming Section 4.3.

4.3 Periodic task scheduling

This section introduces an existing famous scheduling policy, rate monotonic scheduling [Liu and Layland, 1973; Baker, 2003], which broadly applied in real-time control systems. Real-time control systems are mostly characterized by periodic activities, this feature is strictly related to the nature of such a system. In fact, most of them are control systems and therefore their functioning depends on sensors' feedback, low level activities, monitoring and so on. Each task is cyclically triggered at a given sample rate, and it has to perform its activity concurrently with other tasks. In this context, the role of the operating systems is crucial because all of these tasks have individual timing requirements and they have to execute within their deadlines. This chapter focus on the problem of scheduling periodic tasks, and the main basic algorithm developed to cope with these specific issues will be treated in detail. They are rate monotonic, earliest deadline first (EDF), deadline monotonic and finally the EDF version to treat tasks with the deadline less than periods. Each algorithm will be introduced basic concepts, schedulability analysis, and guarantee tests. In order to make the read easy, the basic concepts will be pointed out and a set of the hypothesis will be assumed to simplify the scenario without loss of generality. As a consequence of the foregoing mentioned, the following notations will be introduced:

- Γ denotes a set of periodic tasks;
- τ_i denotes a generic periodic i^{th} task;
- $\tau_{i,j}$ denotes the j^{th} instance of task τ_i;
- $r_{i,j}$ denotes the release time of the j^{th} instance of task τ_i;
- Λ_i denotes the phase of task τ_i. It also represents the release time of the forst instance of the task ($\Lambda_i = r_{i,j}$);
- D_i denotes relative deadline of task τ_i;
- $d_{i,j}$ denotes the absolute deadline of the j^{th} instance of task τ_i. It is also given by $d_{i,j} = \Lambda_i + (j1) \cdot T_i + D_i$;
- $s_{i,j}$ denotes the start time of the j^{th} instance of task τ_i. It represents when task start running;

- $f_{i,j}$ denotes the finishing time of task j^{th} instance of task τ_i. It represents when task stop running.

In order to simplify the analysis, the following hypotheses and assumptions will be considered:

- A1. the instances of a periodic task are activated at a constant rate. The interval, T_i, between two consecutive activations represents the period of the task;
- A2. all instances of a periodic task τ_i have the same worst-case execution time (WCET) C_i;
- A3. all instances of a task have the same relative deadline D_i. All the deadlines are assumed to be equal to the periods T_i;
- A4. there not exist any precedence constraints among the tasks belonging to the task-set Γ;
- A5. each task cannot suspend itself (i.e.: for I/O operations);
- A6. task-set Γ is fully-preemptive;
- A7. the system overhead is negligible.

According with the notation introduced above, a task-set can be summarized as follow:

$$\Gamma = \{\tau(\Lambda_i, T_i, C_i), i = 1, \cdots, n\} \tag{4.1}$$

while arrival times $r_{i,j}$ and relative deadline $d_{i,j}$ of the generic k^{th} instance of the i^{th} task can be easily computed as:

$$\tau_{i,k} = \Lambda_i + (K-1) \cdot T_i \tag{4.2}$$

$$d_{i,k} = r_{i,k} + T_i = \Lambda_i + K \cdot T_i \tag{4.3}$$

4.3.1 Utilization factor

Given a task-set Γ, composed of n tasks, the utilization factor is the fraction of time spent by the CPU to execute the task-set. It is formally defined by

$$U = \sum_{1}^{n} \frac{C_i}{T_i} \tag{4.4}$$

where $\frac{C_i}{T_i}$ denotes the fraction of time spent by the CPU for the execution of task τ_i. The utilization factor can be improved by modifying C_i and T_i but there exists a maximum value of U that, in case of overcame, it yields a not schedulable task-set. This particular value depends on the features of the task-set and the adopted scheduling policy. We can denote with $U_{ub}(\Gamma, A)$ the upper bound of the utilization factor for a given task-set Γ and scheduling algorithm A. Under the particular condition in which $U_{ub} = U_{ub}(\Gamma, A)$, the processor is known to be fully utilized, according to this status any other increase of the computation time of even a single task, caused task-set to be not schedulable.

Given an algorithm A and a task-set Γ, let us to introduce the concept of least upper bound $U_{lub}(A)$ of the processor utilization factor, as the minimum of the utilization factors considering all possible task-set that fully utilize the processor:

$$U_{lub}(A) = min_{\Gamma} U_{ub}(\Gamma, A) \tag{4.5}$$

Since $U_{lub}(A)$ is the minimum of all upper bounds, any task-set with a utilization factor less or equal to $U_{lub}(A)$ is certainly schedulable. Another important result concerning the utilization factor is that a task-set with a utilization factor greater than one, cannot be scheduled by any algorithm:

$$\forall A, \Gamma | U(A) > 1 \Rightarrow \Gamma \text{ is not schedulable by A} \tag{4.6}$$

4.3.2 Rate Monotonic scheduling

Rate Monotonic (RM) is a fixed priority scheduling algorithm that consists of a priority rule assignment based on task arrivals rate. Once the priorities are assigned they cannot be

modified at run-time. Tasks with a high rate of arriving instants have high priorities, tasks that are characterized by a low rate of arrival instants have low priorities. Another feature of RM is that it is intrinsically preemptive because if a new instance of a task arrives and it has a greater priority, it preempts the task currently executing [Liu and Layland, 1973; Baker, 2003].

Under rate monotonic policy [Liu and Layland, 1973], the U_{lub} value calculated for an arbitrary number N of tasks composing the task set is:

$$U_{lub} = n \cdot (2^{1/n} - 1) \tag{4.7}$$

This value decrease with n and for a high value of n, the least upper bound converges to

$$U_{lub}(RM) = ln2 \cong 0.69 \tag{4.8}$$

To check for the schedulability of a given task set under RM, the following condition has to be verified

$$U = \sum_{1}^{n} \frac{C_i}{T_i} < U_{lub}(RM) = n \cdot (2^{1/n} - 1) \tag{4.9}$$

Acording to the above Eq. 4.7 and Eq. 4.9, we can have the rule of rate monotonic scheduling policy in Eq. 4.10, which is applied into the following DVFS-based, energy- and delay-aware traffic management framework.

$$U_{lub}(RM) \leq n \cdot (2^{1/n} - 1) \tag{4.10}$$

4.4 Network and Flow Specification

The proposed energy- and delay-aware traffic management framework assumes the existence of an *edge-controller* that regulates access to the network. The controller relies on the existing routing protocol infrastructure to compute routing paths between a traffic source and destination. Furthermore, the controller uses a flow's traffic specification to verify the

feasibility of the computed routing path to accommodate the flow's QoS requirements. If successful, the router establishes and maintains the selected path to route traffic generated by QoS flows. Otherwise, the flow request is rejected.

The network is defined as a graph $N = (R, L)$, where R represents the set of routers and L the set of links between routers. A router $r \in R$ is characterized by its frequency-dependent execution rate, $\sigma_r^{min} \leq \sigma_r \leq \sigma_r^{max}$, where σ_r^{min} and σ_r^{max} represent the minimum and maximum execution rates, respectively. We use $\mathcal{P}$ to denote the set of paths in the network. A path, $p \in \mathcal{P}$, of length K, is defined as $p = \{r \in R\ (1 \leq r \leq K) \,|\, (r, r+1) \in L\}$. Let $\mathcal{F}$ be the set of traffic flows supported by the network.

A flow, $f \in \mathcal{F}$, is characterized by its end-to-end delay bound, Δ_f, and its traffic rate specification vector, (ρ_f, β_f), where ρ_f represent the f's long-term average packet rate of the flow and β_f its maximum packet burst size. In this chapter, we assume a linear bounded arrival processes (LBAP). Consequently, the maximum number of packets, $\Omega_f(\tau)$, generated by f over a time interval of size τ, does not exceed $\rho_f \cdot \tau + \beta_f, \forall \tau > 0$.

In the following, we formulate the energy- and delay-aware flow establishment problem as an energy minimization problem, subject to end-to-end delay requirements. We then describe a methodology to compute a set of feasible per-router delays along the routing path, to meet end-to-end delay requirements and minimize the path's energy consumption. Finally, we describe an effective strategy to minimize energy consumption under QoS constraints.

4.5 The General Problem Formulation

Definition 1. *A flow, $f \in \mathcal{F}$, characterized by (ρ_f, β_f) and Δ_f, is* **delay-feasible** *over path $p = \{1, \cdots, r, \cdots, K\}$ if and only if $\exists\ \vec{\delta}_f = (\delta_{f,1}, \cdots, \delta_{f,r}, \cdots, \delta_{f,K})$ such that $\sum_{r=1}^{K} \delta_{f,r} \leq \Delta_f$, where $\delta_{f,r}$ represents the delay a packet generated by flow f suffers at router $r (1 \leq r \leq K)$ along path p.*

Definition 2. *Let $\mathcal{F}_r \subset \mathcal{F}$ ($F_r = \|\mathcal{F}_r\|$) represent the set of flows traversing router r. Router r is said to be* **$\mathcal{F}_r$-feasible** *if $\forall f \in \mathcal{F}_r$, f is delay-feasible over its routing path, p_f.*

Definition 3. *A path $p = \{1, \cdots, r, \cdots, K\}$ is said to be* **energy-optimum** *if $\forall r \in p$, r is $\mathcal{F}_r$-feasible and the energy consumed by r, $\mathcal{E}_r = \sum_{f \in \mathcal{F}_r} \mathcal{E}_f$, is minimum, where $\mathcal{E}_f$ is the energy consumed by flow $f \in \mathcal{F}_r$.*

Let $\mathcal{F}_p \subset \mathcal{F}$ be the set of flows traversing path p ($F_p = \|\mathcal{F}_p\|$). Path p is energy-optimal if there exists $\vec{\sigma}_p = (\sigma_1, \cdots, \sigma_r, \cdots, \sigma_K)$ ($\sigma_r \in [\sigma_r^{min}, \sigma_r^{max}]$) and $\vec{\delta}_f = (\delta_{f,1}, \cdots, \delta_{f,r}, \cdots, \delta_{f,K})$ such that (i) $\forall f \in \mathcal{F}_p$, f is delay-feasible over p, (ii) $\forall r \in p$, r is $\mathcal{F}_r$-feasible, and (iii) the energy consumed by p, $\mathcal{E}_p = \sum_{r \in p} \mathcal{E}_r$, is minimum. Consequently, for a given path, $p \in \mathcal{P}$, the energy-aware and delay-assignment flow establishment problem can be formulated as follows:

$$
\begin{aligned}
\textbf{Minimize} \quad & \mathcal{E}_p(\vec{\sigma}_p, \vec{\delta}_f) \\
\textbf{Subject to} \quad & l_{f,r} \leq \delta_{f,r} \leq h_{f,r}, \ (1 \leq r \leq K) \text{ and } (f \in \mathcal{F}_p) \\
& \sigma_r^{min} \leq \sigma_r \leq \sigma_r^{max}, \ (1 \leq r \leq K) \\
& \sum_{r=1}^{K} \delta_{f,r} \leq \Delta_f, \ (f \in \mathcal{F}_p)
\end{aligned}
$$

where:

- $l_{f,r}$: a lower bound on the delay values router, r, can assign to flow, f, traversing path, p,
- $h_{f,r}$: an upper bound on the delay values router, r, can assign to flow, f, traversing path, p,
- Δ_f: the end-to-end delay bound for flow, f, traversing path, p,
- σ_r : the execution rate at router, $r \in p$, and
- σ_r^{min} and σ_r^{max}: the minimum and maximum execution rates of router, r. These rates are dependent on the associated operational frequencies, ψ_r^{min} and ψ_r^{max} at router, r.

Consider an energy-optimal path, p, supporting a flow set $\mathcal{F}_p$. In order to assess the feasibility of accepting a new flow, n, each router r across p must determine the energy-optimum execution rate, σ_r^*, and a delay, $\delta_{n,r}$, to meet the end-to-end delay requirement of the new flow, without violating the delay requirements of the currently supported flows in $\mathcal{F}_p$. In the following, we first introduce the delay-based scheduling policy scheme used by

routers to service packets. We then describe a methodology that can be used to compute, for a given flow, a feasible range of delays for each router along the path. The delay range can be used to assign a per-router delay that guarantees the end-to-end delay requirement of the new flow, while minimizing energy across the path.

4.5.1 Delay-based Packet Scheduling Policy

In the proposed framework, routers use a nonpreemptive delay-based scheduling policy, whereby flows with shorter delays are assigned higher priorities than those with longer delays [Liu and Layland, 1973]. A delay-based scheduling policy is optimal among fixed-priority scheduling algorithms and adheres to flow specification of the DiffServ service model [Chan et al., 2003]. According to the rule of the scheduling policy shown in Eq. 4.10, assuming that a delay-based router r processes packets at a service rate, $\mu_r(\sigma_r)$, a set of flows, $\mathcal{F}_r$ ($F_r = \|\mathcal{F}_r\|$), where each flow f is characterized by $\Omega_f(\delta_{f,r})$ and Δ_f, is delay feasible at router r if the following holds:

$$\sum_{f=1}^{F_r} \frac{W_{f,r}(\sigma_r, \delta_{f,r})}{\delta_{f,r}} \leq U(F_r) - \frac{\overline{u}}{\underline{\Delta}}, \tag{4.11}$$

where $W_{f,r}(\sigma_r, \delta_{f,r}) = \frac{\Omega_f(\delta_{f,r})}{\mu_r(\sigma_r)}$, represents the maximum amount of service time required to process packets generated by flow f, at router, r, over a time interval of size $\delta_{f,r}$. The term $\frac{\overline{u}}{\underline{\Delta}}$, $\overline{u} = max_{1 \leq f \leq F_r} \{u_{f,r}\}$, where $u_{f,r} = \frac{1}{\mu_r(\sigma_r)}$ denotes the service time used to process a packet from flow f at router, r, and $\underline{\Delta} = min_{1 \leq f \leq F_r} \{\delta_{f,r}\}$ accounts for the nonpreemptive aspect of the scheduling policy at router, r. It represents the maximum amount of time a higher priority packet, arriving just at the instant a lower priority packet gained access to the server, may be forced to wait before being serviced by r [Znati and Melhem, 2004; Field et al., 1995]. $U(F_r)$ denotes the total percentage of r's processing capacity which can be allocated to provide guaranteed service to the flow set, $\mathcal{F}_r$. For a delay-based scheduling policy, $U(F_r) = F_r \cdot (2^{\frac{1}{F_r}} - 1)$ [Liu and Layland, 1973; Znati and Melhem, 2004]. In the following, we describe a methodology used to compute a feasible delay range to a new flow at a given router along a routing path.

4.5.2 Per-router Delay Computation

The characterization of the processing capacity and the flow traffic load provide a basis for the computation of the smallest and largest per-router delay bounds that can be assigned by a router to a new flow [Znati and Melhem, 2004].

4.5.2.1 Smallest Feasible Delay Let $\mathcal{F}_r$ ($\|\mathcal{F}_r\| = F_r$) represent the flow set currently supported by a delay-based nonpreemptive router, r, executing at a rate σ_r. The exact criterion for a new flow n to be delay-feasible over path p, without violating current flows delay requirements, can be expressed as:

$$\frac{W_{n,r}(\sigma_r^*, \delta_{n,r})}{\delta_{n,r}} + \sum_{f=1}^{F_r} \frac{W_{f,r}(\sigma_r^*, \delta_{f,r})}{\delta_{f,r}} \leq U(F_r^+) - \frac{\overline{u}}{min\,(\underline{\Delta}, \delta_{n,r})} \tag{4.12}$$

where $\mathcal{F}_r^+ = \mathcal{F}_r \cup \{n\}$ ($F_r^+ = \|\mathcal{F}_r^+\| = F_r + 1$) represents the new flow set supported by r, executing at the new rate, σ_r^*, and $\delta_{n,r}$ represents the delay flow n's packets suffer at router r.

The value $\delta_{n,r} = l_{n,r}$, for which the equality holds, specifies a lower bound on the delay values router, r, can offer to flow, n, based on r's current processing excess capacity. This smallest feasible delay value is achieved by dedicating all router r's excess processing capacity to flow, $n \in \mathcal{F}_r^+$, and can be further derived as:

$$l_{n,r} = \begin{cases} \dfrac{\beta_n + 1}{\mu_r(\sigma_r^*) \cdot U(F_r^+) - \sum_{f=1}^{F_r} \frac{\Omega_f(\delta_{f,r})}{\delta_{f,r}} - \rho_n}, \\ \qquad\qquad \text{if } \delta_{n,r} \leq \underline{\Delta} \\ \dfrac{\beta_n}{\mu_r(\sigma_r^*) \cdot U(F_r^+) - \sum_{f=1}^{F_r} \frac{\Omega_f(\delta_{f,r})}{\delta_{f,r}} - \frac{1}{\underline{\Delta}} - \rho_n}, \\ \qquad\qquad \text{if } \delta_{n,r} > \underline{\Delta} \end{cases} \tag{4.13}$$

4.5.2.2 Largest Feasible Delay The maximum feasible router delays of given traffic flow are correlated with its end-to-end delay requirement. Consequently, the upper bound on the delay value, $h_{n,r}$, a router r ($1 \leq r \leq K$) can assign to a new flow, n, must verify $\sum_{r=1}^{K} h_{n,r} = \Delta_n$. For a given routing path, p of length K, the largest feasible delay value,

$h_{n,r}$, assigned to flow, n by router, r, can be expressed as $h_{n,r} = \zeta_r \cdot \Delta_n$. Assuming that the routers across path p are uniformly loaded, ζ_r can be set to $\frac{1}{K}$. If the load across the routers is unevenly distributed, however, ζ_r can be set to $\frac{\vartheta_r}{\sum_{r \in p} \vartheta_r}$, where ϑ_r represents the load at router, r. The second upper bound values assignment will be further discussed in Section 4.5.7.2.

In the following, we discuss the model used to derive power and energy consumption. We then formalize the energy- and delay-aware minimization problem to reduce energy consumption, while adhering to flows' delay requirements.

4.5.3 Power Model

Routers in large scale networks are typically equipment with specialized ASIC hardware to handle most of the data plane traffic processing and forwarding tasks. These processors are generally among the most energy-consuming components of the router [Chabarek et al., 2008]. The execution rate, σ, of a DVFS-enable processor, with a minimum frequency, ψ^{min}, and a maximum frequency, ψ^{max}, ranges from σ^{min} to σ^{max}. Therefore, the dynamic power consumption of a computing router executing at rate, σ, can be approximately expressed as $\varphi^D(\sigma) = \alpha \cdot \sigma^3$, where α is a constant [Yu et al., 2015b,a]. In addition to the load-dependent dynamic power, the bias and leakage current to support the execution of load-independent control and data plane tasks contribute to the static power consumption, which is independent of the processor rate [Vishwanath et al., 2014; Yu et al., 2015a]. In this chapter, we define the static power ratio, ω, as a fixed fraction of the router power consumed when executing at maximum rate [Cui et al., 2014]. Hence, the power consumption of an active router processor can be expressed as: $\varphi(\sigma) = \omega \cdot \alpha \cdot (\sigma^{max})^3 + (1-\omega) \cdot \alpha \cdot \sigma^3$. Typically, the dynamic power constitutes up to 30% of the total power, resulting in $\omega \geq 0.7$ [Imaizumi and Morikawa, 2010; Wobker, 2012].

4.5.4 Router-based Energy Consumption Model

Consider a set of flows, $\mathcal{F}_r \subseteq \mathcal{F}$ ($F_r = \|\mathcal{F}_r\|$), currently supported by router, $r \in R$. Each flow, f in $\mathcal{F}_r$ is characterized by its per-router delay, $\delta_{f,r}$. Furthermore, assume that router,

r, operating at a feasible execution rate, $\sigma_r^{min} \leq \sigma_r \leq \sigma_r^{max}$, can process all flows in $f \in \mathcal{F}_r$, without violating their end-to-end delay requirements. The dynamic energy, $\mathcal{E}_r^D(\sigma_r, \vec{\delta}_r|_{\forall f \in \mathcal{F}_r})$ consumed by r to process packets generated by all flows, $1 \leq f \leq F_r$, can be expressed as:

$$\begin{aligned} \mathcal{E}_r^D(\sigma_r, \vec{\delta}_r|_{\forall f \in \mathcal{F}_r}) &= \varphi^D(\sigma_r) \cdot \sum_{f=1}^{F_r} W_{f,r}(\sigma_r, \delta_{f,r}) \\ &= \theta_r \cdot \left(\sum_{f=1}^{F_r} \Omega_f(\delta_{f,r}) \right) \cdot (\sigma_r)^2 \end{aligned} \tag{4.14}$$

where $\vec{\delta}_r|_{\forall f \in \mathcal{F}_r}$ represents the per-router delay vector, $(\delta_{1,r}, \cdots, \delta_{f,r}, \cdots, \delta_{F_r,r})$ and $\theta_r = \alpha_r \cdot IPP_r$ for router, r. Assuming IPP_r represents the number of instructions to complete the processing and transmission of a packet, at router, r, is:

$$W_{f,r}(\sigma_r, \delta_{f,r}) = \frac{\Omega_f(\delta_{f,r})}{\mu_r(\sigma_r)} = \frac{IPP_r \cdot (\rho_f \cdot \delta_{f,r} + \beta_f)}{\sigma_r} \tag{4.15}$$

4.5.5 Path-based Energy Consumption Model

Consider a routing path, $p \in \mathcal{P}$ of length K, where each router r supports a set of flows $\mathcal{F}_r (1 \leq r \leq K)$. Let $\mathcal{F}_c = \{\mathcal{F}_1, \cdots, \mathcal{F}_r, \cdots, \mathcal{F}_K\}$ represent the super set of flows supported along path, p. Furthermore, let $\vec{\sigma}_p = (\sigma_1, \cdots, \sigma_r, \cdots, \sigma_K)$ represent the feasible execution rate vector of the routers along path, p, and $\vec{\delta} = \left(\vec{\delta}_1|_{\forall f \in \mathcal{F}_1}, \cdots, \vec{\delta}_r|_{\forall f \in \mathcal{F}_r}, \cdots, \vec{\delta}_K|_{\forall f \in \mathcal{F}_K}\right)$ represent the feasible delays for the flow set, $\mathcal{F}_c$. Thus, the energy consumed by processing all currently scheduled flows, $\mathcal{F}_c$, over their feasible delays, $\vec{\delta}$, can be further derived as:

$$\begin{aligned} \mathcal{E}_p^D(\vec{\sigma}_p, \vec{\delta}) &= \sum_{r=1}^{K} \mathcal{E}_r^D(\sigma_r, \vec{\delta}_r|_{\forall f \in \mathcal{F}_r}) \\ &= \sum_{r=1}^{K} \theta_r \cdot \left(\sum_{f=1}^{F_r} \Omega_f(\delta_{f,r}) \right) \cdot (\sigma_r)^2 \end{aligned} \tag{4.16}$$

where $F_r = \|\mathcal{F}_r\|$, $1 \leq r \leq K$.

Consider a new flow, n, traversing path, p, and let $\mathcal{F}_c^+ = \left\{\mathcal{F}_1^+, \cdots, \mathcal{F}_r^+, \cdots, \mathcal{F}_K^+\right\}$ represent a new set of schedulable flows across p, where $\mathcal{F}_r^+ = \mathcal{F}_r \cup n$ $(1 \leq r \leq K)$. Furthermore, let $\vec{\sigma}_p^* = (\sigma_1^*, \cdots, \sigma_r^*, \cdots, \sigma_K^*)$, be the new execution rate vector required to achieve a delay

vector $\vec{\delta}_n = (\delta_{n,1}, \cdots, \delta_{n,r}, \cdots, \delta_{n,K})$ that meets n's end-to-end requirement. The total energy consumed by processing packets generated by flows, $\mathcal{F}_c^+$, over their respective feasible delays, $\vec{\delta}^+ = \left(\vec{\delta}_1|_{\forall f \in \mathcal{F}_1^+}, \cdots, \vec{\delta}_r|_{\forall f \in \mathcal{F}_r^+}, \cdots, \vec{\delta}_K|_{\forall f \in \mathcal{F}_K^+}\right)$, can be expressed as:

$$\begin{aligned} \mathcal{E}_p^D(\vec{\sigma}_p^*, \vec{\delta}^+) &= \underbrace{\mathcal{E}_p^D(\vec{\sigma}_p^*, \vec{\delta})}_{\forall f \in \mathcal{F}_c} + \underbrace{\mathcal{E}_p^D(\vec{\sigma}_p^*, \vec{\delta}_n)}_{n} \\ &= \sum_{r=1}^{K} \theta_r \cdot \left(\Omega_n(\delta_{n,r}) + \sum_{f=1}^{F_r} \Omega_f(\delta_{f,r}) \right) \cdot (\sigma_r^*)^2 \end{aligned} \tag{4.17}$$

where $\Omega_n(\delta_{n,r}) = \rho_n \cdot \delta_{n,r} + \beta_n$. Note that $\sum_{f=1}^{F_r} \Omega_f(\delta_{f,r})$ is independent of the $\sigma_r^* (1 \leq r \leq K)$, Eq. 4.17 reduces to:

$$\mathcal{E}_p^D(\vec{\sigma}_p^*, \vec{\delta}_n) = \sum_{r=1}^{K} (a_{n,r} \cdot \delta_{n,r} + b_{n,r}) \cdot (\sigma_r^*)^2 \tag{4.18}$$

where $a_{n,r} = \theta_r \cdot \rho_n$ and $b_{n,r} = \theta_r \cdot \left(\beta_n + \sum_{f=1}^{F_r} \Omega_f(\delta_{f,r})\right)$. Following, we formalize the energy-aware, delay-assignment problem.

4.5.6 Energy- and Delay-aware Flow Scheduling

Assume the network receives a request to establish a new flow, characterized by its traffic rate specification vector (ρ, β) and its end-to-end delay value Δ, over a path, p. A feasible solution to minimize energy, while adhering to end-to-end delay requirements, must achieve the following requirements.

- The per-router delay assignments are feasible across the routing path;
- The per-router execution rate assignments are feasible across the routing path;
- The end-to-end delay requirements of the new flow are enforced without violating the delay requirements of the currently supported flows; and
- The energy consumed by routers across the path is minimum.

Since the static power is independent of the traffic workload, only the load-dependent energy is considered in our optimization objective to determine the new optimal routers' execution rates vector, $\vec{\sigma} = (\sigma_1, \cdots, \sigma_K)$ and the per-router delay vector, $\vec{\delta} = (\delta_1, \cdots, \delta_K)$

of the new flow, across the routing path p. Given that the workload requested by the new flow at router, r, over a time interval, δ_r, is $\rho \cdot \delta_r + \beta$, the optimization problem is reduced to minimizing the dynamic energy consumption of the new and the currently supported flows across the routing path, p, while adhering to delay requirements of the supported flows. Therefore, the objective function can be expressed as:

$$\mathcal{E}_p^D(\vec{\sigma}, \vec{\delta}) = \sum_{r=1}^{K} (a_r \cdot \delta_r + b_r) \cdot (\sigma_r)^2 \tag{4.19}$$

It is worth noting that the sum of per-router delays across the routing path must approach as closely as possible the new flow requested end-to-end delay budget, Δ, in order to optimize the objective function. Thus, the per-router delay and per-router execution rate assignment that minimizes energy reduces to finding $\vec{\sigma}$ and $\vec{\delta}$ such that $\sum_{r=1}^{K}(a_r \cdot \delta_r + b_r) \cdot (\sigma_r)^2$ is minimum, where a_r and b_r, $1 \leq r \leq K$, are two constants defined in Eq. 4.18. Furthermore, a solution must satisfy the delay constraints, namely $\sum_{r=1}^{K} \delta_r \leq \Delta, l_r \leq \delta_r \leq h_r$, and $\sigma_r^{min} \leq \sigma_r \leq \sigma_r^{max}$, where l_r and h_r represent the lower and upper delay bounds values at router, r, respectively, and $[\sigma_r^{min}, \sigma_r^{max}]$ denote the range of per-router execution rates of router, r $(1 \leq r \leq K)$.

To minimize energy consumption, while adhering to flows' end-to-end delay requirements, the *Energy- and Delay-aware Flow Scheduling* (EDFS) optimization problem can be formalized as:

$$\begin{aligned}
\textbf{minimize} \quad & \mathcal{E}_p^D(\vec{\sigma}, \vec{\delta}) = \sum_{r=1}^{K} (a_r \cdot \delta_r + b_r) \cdot (\sigma_r)^2 \\
\textbf{subject to} \quad & g_r(\vec{\sigma}, \vec{\delta}) = l_r - \delta_r \leq 0, \ (1 \leq r \leq K) \\
& g_{K+r}(\vec{\sigma}, \vec{\delta}) = \delta_r - h_r \leq 0, \ (1 \leq r \leq K) \\
& g_{2K+r}(\vec{\sigma}, \vec{\delta}) = \sigma_r^{min} - \sigma_r \leq 0, \ (1 \leq r \leq K) \\
& g_{3K+r}(\vec{\sigma}, \vec{\delta}) = \sigma_r - \sigma_r^{max} \leq 0, \ (1 \leq r \leq K) \\
& d(\vec{\sigma}, \vec{\delta}) = \sum_{r=1}^{K} \delta_r - \Delta = 0
\end{aligned}$$

Note that if $\sum_{r=1}^{K} l_r \geq \Delta$, then no per-node delay assignment is feasible and the request for the flow establishment should be rejected. Furthermore, if $\sum_{r=1}^{K} l_r = \Delta$, then the optimal

solution is to set $\delta_r = l_r$, for all routers across the path. This is due to the fact that increasing δ_r, to slowdown the processor and save energy, causes the violation of the flow's end-to-end delay. It is also clear that the flow's end-to-end delay, Δ, should be used in its entirety in order to optimize the objective function. This stems from the observation that if there exists a set of delay values, $l_r \leq \delta_r \leq h_r (1 \leq r \leq K)$, such that $\sum_{r=1}^{K} \delta_r < \Delta$, it is easy to show that there exist a set of $\hat{\delta}_r$ and a set of ϵ_r, such that $\hat{\delta}_r = \delta_r + \epsilon_r (1 \leq r \leq K; \epsilon_r > 0)$, $\sum_{r=1}^{K} \hat{\delta}_r = \hat{\Delta}$; and $\hat{\Delta}_r (1 \leq r \leq K)$ further minimize energy consumption, thereby negating the optimality of δ_r's. Based on this observation and using the fact that $\delta_r \geq l_r$, for all $r = 1, \cdots, K$, the per-router delay optimization problem can be expressed as: Minimize $\mathcal{E}_p^D(\vec{\sigma}, \vec{\hat{\delta}}) = \sum_{r=1}^{K} (a_r \cdot (l_r + \hat{\delta}_r) + b_r) \cdot (\sigma_r)^2$, subject to: $\sum_{r=1}^{K} \hat{\delta}_r = \hat{\Delta}$, $0 \leq \hat{\delta}_r$ and $\hat{\delta}_r \leq \hat{h}_r = h_r - l_r$, where $\hat{\Delta}_r = \Delta - \sum_{r=1}^{K} l_r$, $\hat{\delta}_r = \delta_r - l_r$ and $\hat{h}_r = h_r - l_r$. To solve the EDFS optimization problem, approaches described in [Znati and Melhem, 2004]. It is to be noted that in the above formulation, $\mathcal{E}_p^D$, $g_i (i : 1, \cdots, 4K)$, and $d()$ are convex. The Kuhn-Tucker conditions of optimality conditions states that a solution $\left(\vec{\sigma}, \vec{\hat{\delta}}\right)$ to the above problem is globally optimal if and only if there exist a scalar $\lambda_j \geq 0$ for $j \in I = \left\{ j : g_j(\vec{\sigma}, \vec{\hat{\delta}}) = 0 \right\}$, and a scalar υ such that: $\nabla \mathcal{E}_p^D(\vec{\sigma}, \vec{\hat{\delta}}) + \sum_{j \in I} \lambda_j \cdot \nabla g_j(\vec{\sigma}, \vec{\hat{\delta}}) + \upsilon \cdot \nabla d(\vec{\sigma}, \vec{\hat{\delta}}) = 0$. Based on the observation, the EDFS problem, referred to as *Opt_EDFS*, can be solved by first solving the auxiliary optimization problem, referred to as *Opt_ED*, which considers only the equality constraint of delays. Then a second problem, referred to as *Opt_LD*, which takes into consideration the equality and lower bound constraints of delays, but ignores their upper bound constraints, is solved. The solutions to these problems can be used to solve the original EDFS optimization problem.

4.5.6.1 ***Opt_ED* Solution** The *Opt_ED* problem does not take into account the boundary constraints of delays, and thus can be expressed as:

$$\textbf{minimize} \quad \mathcal{E}_p^D(\vec{\sigma}, \vec{\hat{\delta}}) = \sum_{r=1}^{K} (a_r \cdot (l_r + \hat{\delta}_r) + b_r) \cdot (\sigma_r)^2 \tag{4.20}$$

$$\textbf{subject to} \quad \sigma_r^{min} - \sigma_r \leq 0, \ (1 \leq r \leq K) \tag{4.21}$$

$$\sigma_r - \sigma_r^{max} \leq 0,\ (1 \leq r \leq K) \tag{4.22}$$

$$\sum_{r=1}^{K} \hat{\delta}_r - \hat{\Delta} = 0 \tag{4.23}$$

The application of Lagrange multipliers technique to the above problem yields

$$a_r \cdot (\sigma_r)^2 + \upsilon = 0,\ r = 1, \cdots, K \tag{4.24}$$

$$\begin{aligned} &2 \cdot \sigma_r \cdot (a_r \cdot (l_r(\sigma_r) + \hat{\delta}_r) + b_r) - Z_r = 0, \\ &where\ Z_r = \lambda_{2K+r} - \lambda_{3K+r},\ r = 1, \cdots, K \end{aligned} \tag{4.25}$$

$$\lambda_{2K+r} \cdot (\sigma_r^{min} - \sigma_r) = 0,\ r = 1, \cdots, K \tag{4.26}$$

$$\lambda_{3K+r} \cdot (\sigma_r - \sigma_r^{max}) = 0,\ r = 1, \cdots, K \tag{4.27}$$

$$\lambda_{2K+r}, \lambda_{3K+r} \geq 0,\ r = 1, \cdots, K \tag{4.28}$$

where $\lambda_j, j = 2K+1, \cdots, 4K$ and υ are the Lagrange multipliers. Using the fact that $\sum_{r=1}^{K} \hat{\delta}_r = \hat{\Delta}$,$\hat{\delta}_r = \delta_r - l_r$, and Eq. 4.24∼4.28, results in

$$\sigma_r = \sqrt{\frac{|\upsilon|}{a_r}},\ r = 1, \cdots, K \tag{4.29}$$

$$\begin{aligned} \delta_r &= \frac{\lambda_{2k+r} - \lambda_{3K+r}}{2\sqrt{a_r \cdot |\upsilon|}} - \frac{b_r}{a_r},\ r = 1, \cdots, K \\ \hat{\delta}_r &= \delta_r - l_r(\sigma_r = \sqrt{\frac{|\upsilon|}{a_r}}),\ r = 1, \cdots, K \end{aligned} \tag{4.30}$$

According to Eq. 4.13, $l_r(\sigma_r = \sqrt{\frac{|\upsilon|}{a_r}})$ in Eq. 4.30 can be expressed as:

$$l_r(\sigma_r = \sqrt{\frac{|\upsilon|}{a_r}}) = \begin{cases} \dfrac{C_2}{C_1 \cdot \sqrt{\frac{|\upsilon|}{a_r}} - C_3}, \text{ if } l_r \leq \underline{\Delta} \\ \dfrac{C_4}{C_1 \cdot \sqrt{\frac{|\upsilon|}{a_r}} - C_5}, \text{ if } l_r > \underline{\Delta} \end{cases} \tag{4.31}$$

where $C_1 = \frac{U(F_r^+)}{IPP_r}$, $C_2 = \beta + 1$, $C_3 = \sum_{f=1}^{F_r} \frac{\Omega_f(\delta_{f,r})}{\delta_{f,r}} - \rho$, $C_4 = \beta$ and $C_5 = C_3 + \frac{1}{\underline{\Delta}}$.

Lemma 1. *If Opt_ED violates some inequality constraints given by Eq. 4.21 and Eq. 4.22, then $\exists r$ such that $\lambda_{2k+r} = 0$ or $\lambda_{2k+r} \cdot \lambda_{3k+r} > 0$.*

Proof. Assume that $\exists r$, $\lambda_{2k+r} = 0$. In this case, Eq. 4.25 implies that $\exists r$, $\hat{\delta}_r < 0$ (or $\delta_r < 0$) as shown in Eq. 4.30, this violates the optimality property of the solution. Therefore, $\forall r$, the Lagrange multiplier $\lambda_{2k+r}, 1 \leq r \leq K$ are strictly greater than 0. Furthermore, assume $\forall r$, $\lambda_{2k+r} > 0$ and $\exists r$, $\lambda_{3K+r} > 0$. In this case, Eq. 4.26 and Eq. 4.27 imply that $\exists r$, $\sigma_r = \sigma_r^{min}$ and $\sigma_r = \sigma_r^{max}$ at the same time, which results in the occurrence of inequality constraint violations in Eq. 4.21 and Eq. 4.22. Therefore, $\forall r$, λ_{3k+r} is strictly equal to 0. □

Lemma 2. *If Opt_ED violates some inequality constraints given by Eq. 4.23, then $\exists r$ such that $\lambda_{2k+r} \leq 2\sigma_r^{min} \cdot (a_r \cdot l_r(\sigma_r^{min}) + b_r)$, where $\upsilon = -a_r \cdot (\sigma_r^{min})^2$, $\lambda_{3k+r} = 0$ according to Lemma 1.*

Proof. Assume that $\forall r$, $\lambda_{2k+r} \leq 2\sigma_r^{min} \cdot (a_r \cdot l_r(\sigma_r^{min}) + b_r)$. In this case, Eq. 4.30 implies that $\forall r$, $\hat{\delta}_r \leq 0$. This implies that $\sum_{r=1}^{K} \hat{\delta}_r$ is less than or equal to 0, and the delay budget, $\hat{\Delta}$, remains totally unused. This violates the optimality property of the solution. □

Hence, if there were a solution to *Opt_ED* where for $\forall r$, $\lambda_{2k+r} > 0$ and $\lambda_{3K+r} = 0$, then the solution will be discovered by solving a set of nonlinear equations which are identical to Kuhn-Tucker conditions. In this solution, since $\forall r$, $\sigma_r = \sigma_r^{min}$, *Opt_ED* is an optimal solution for the case that the maximal traffic load along the given path can be accepted under the constraints of $\sigma_r = \sigma_r^{min}$ and $\sum_{r=1}^{K} \hat{\delta}_r = \hat{\Delta}$. In other words, *Opt_ED* algorithm is fit for minimizing energy consumption for the maximal acceptance of traffic load under the minimal execution rates of the routers along the path. If the traffic load is far from this the maximal acceptance value, the constraint Eq. 4.23 should be replaced by $\sum_{r=1}^{K} \hat{\delta}_r - \hat{\Delta} \leq 0$ to

find optimal per-router delay assignment for the low traffic load. From the above observation and discussion, *Opt_ED* would not achieve the goal of significant energy saving through dynamical speed scaling to adapt to the higher traffic load. The following optimization algorithm *Opt_LD* is used to further explore this issue.

4.5.6.2 *Opt_LD* Solution

The *Opt_LD* problem can be expressed as

$$\textbf{minimize} \quad \mathcal{E}_p^D(\vec{\sigma}, \vec{\hat{\delta}}) = \sum_{r=1}^{K} (a_r \cdot (l_r + \hat{\delta}_r) + b_r) \cdot (\sigma_r)^2 \tag{4.32}$$

$$\textbf{subject to} \quad 0 - \hat{\delta}_r \leq 0, \ (1 \leq r \leq K) \tag{4.33}$$

$$\sigma_r^{min} - \sigma_r \leq 0, \ (1 \leq r \leq K) \tag{4.34}$$

$$\sigma_r - \sigma_r^{max} \leq 0, \ (1 \leq r \leq K) \tag{4.35}$$

$$\sum_{r=1}^{K} \hat{\delta}_r - \hat{\Delta} = 0 \tag{4.36}$$

To solve *Opt_LD*, we first evaluate the solution set S_{Opt_LD} to the corresponding problem *Opt_ED* and check whether all inequality constraints are automatically satisfied. If this is the case, the solution set of the *Opt_LD* problem reduces to the solution set, S_{Opt_LD}, which can be used to find the traffic maximal acceptance under the minimal execution rates, while minimizing the corresponding energy consumption. Otherwise, S_{Opt_LD}, especially for more new traffic flows' requests, will be constructed iteratively as described below.

A well-known result of nonlinear optimization theory states that the solution S_{Opt_LD} of the *Opt_LD* must satisfies Kuhn-Tucker conditions [Panik, 1976]. Furthermore, Kuhn-Tucker conditions are also sufficient due to the properties of the objective function. For Problem *Opt_LD*, Kuhn-Tucker conditions can be derived from Eq. 4.33$\sim$ Eq. 4.36 as

$$a_r \cdot (\sigma_r)^2 + \upsilon - \lambda_r = 0, \ r = 1, \cdots, K \tag{4.37}$$

$$\lambda_r \cdot \frac{\partial l_r(\sigma_r)}{\partial \sigma_r} + 2 \cdot \sigma_r \cdot (a_r \cdot (l_r(\sigma_r) + \hat{\delta}_r) + b_r) - Z_r = 0, \\ where\ Z_r = \lambda_{2K+r} - \lambda_{3K+r},\ r = 1, \cdots, K \tag{4.38}$$

$$-\lambda_r \cdot \hat{\delta}_r = 0,\ r = 1, \cdots, K \tag{4.39}$$

$$\lambda_{2K+r} \cdot (\sigma_r^{min} - \sigma_r) = 0,\ r = 1, \cdots, K \tag{4.40}$$

$$\lambda_{3K+r} \cdot (\sigma_r - \sigma_r^{max}) = 0,\ r = 1, \cdots, K \tag{4.41}$$

$$\lambda_r, \lambda_{2K+r}, \lambda_{3K+r} \geq 0,\ r = 1, \cdots, K \tag{4.42}$$

where $\lambda_j, j = 1, \cdots, K, 2K+1, \cdots, 4K$ and υ are the Lagrange multipliers. The necessary and sufficient character of Kuhn-Tucker conditions provides optimal values for *Opt_LD*. One method for solving the optimization problem *Opt_LD* is to find a solution to Eq. 4.37, Eq. 4.38, Eq. 4.40 and Eq. 4.41 which satisfies constraint sets Eq. 4.39 and Eq. 4.42. Iteratively solving the nonlinear equations is a complex process that is not guaranteed to converge. A more efficient approach to the solution uses the Kuhn-Tucker conditions Eq. 4.37$\sim$ Eq. 4.42 to prove some useful properties of the optimal solution. The properties derived are then used to refine the solution of the optimization problem *Opt_ED*. Using the fact that $\sum_{r=1}^{K} \hat{\delta}_r = \hat{\Delta}$, $\hat{\delta}_r \geq 0$, and Eq. 4.37$\sim$ Eq. 4.42, results in

$$\sigma_r = \sqrt{\frac{|\lambda_r - \upsilon|}{a_r}},\ r = 1, \cdots, K \tag{4.43}$$

$$\delta_r = \frac{\lambda_{2k+r} - \lambda_{3K+r} - \lambda_r \cdot \frac{\partial l_r(\sigma_r)}{\partial \sigma_r}\Big|_{\sigma_r = \sqrt{\frac{|\lambda_r - \upsilon|}{ar}}}}{2\sqrt{a_r \cdot |\lambda_r - \upsilon|}} - \frac{b_r}{a_r}, \\ r = 1, \cdots, K \\ \hat{\delta}_r = \delta_r - l_r(\sigma_r = \sqrt{\frac{|\lambda_r - \upsilon|}{a_r}}),\ r = 1, \cdots, K \tag{4.44}$$

According to Eq. 4.13, $l_r(\sigma_r)$ in Eq. 4.44 can be expressed as:

$$l_r(\sigma_r) = \begin{cases} \dfrac{C_2}{C_1 \cdot \sigma_r - C_3}, & \text{if } l_r \leq \underline{\Delta} \\ \dfrac{C_4}{C_1 \cdot \sigma_r - C_5}, & \text{if } l_r > \underline{\Delta} \end{cases} \tag{4.45}$$

where $C_1 = \frac{U(F_r^+)}{IPP_r}$, $C_2 = \beta + 1$, $C_3 = \sum_{f=1}^{F_r} \frac{\Omega_f(\delta_{f,r})}{\delta_{f,r}} - \rho$, $C_4 = \beta$ and $C_5 = C_3 + \frac{1}{\underline{\Delta}}$.

From Eq. 4.45, we can further derive

$$\frac{\partial l_r(\sigma_r)}{\partial \sigma_r}\Big|_{\sigma_r = \sqrt{\frac{|\lambda_r - \upsilon|}{a_r}}} = \begin{cases} -\dfrac{C_1 \cdot C_2}{(C_1 \cdot \sqrt{\frac{|\lambda_r - \upsilon|}{a_r}} - C_3)^2}, & \text{if } l_r \leq \underline{\Delta} \\ -\dfrac{C_1 \cdot C_4}{(C_1 \cdot \sqrt{\frac{|\lambda_r - \upsilon|}{a_r}} - C_5)^2}, & \text{if } l_r > \underline{\Delta} \end{cases} \tag{4.46}$$

Lemma 3. *If Opt_LD violates some inequality constraints given by Eq. 4.33, then $\exists r$ such that $\lambda_r > 0$.*

Proof. Assume to the contrary that $\forall r$, $\lambda_r = 0$. In this case, Kuhn-Tucker conditions reduce to the equality constraints of *Opt_ED*, the set of inequality constraints Eq. 4.33 plus the Lagrangian condition given in Eq. 4.44. On the other hand, the set should satisfy Eq. 4.36 and the Lagrangian condition Eq. 4.37 and Eq. 4.38. In other words, solving *Opt_ED* is always equivalent to solving a set of nonlinear equations which are identical to Kuhn-Tucker conditions of *Opt_LD*, except for the inequality constraints, by setting $\lambda_r = 0$, $\forall r$. Hence, if there were a solution to *Opt_LD* where for all $\lambda_r = 0$, then the solution will be discovered by *Opt_ED* algorithm described above without the occurrence of any inequality constraint violations. This is in contradiction with the assumption that the solution S_{Opt_ED} fails to satisfy all the inequality constraints. Therefore, there exists at least one Lagrange multiplier λ_r that is strictly greater than 0. □

Lemma 4. *If Opt_LD violates some inequality constraints given by Eq. 4.33, then $\exists r$ such that $\lambda_r = 0$.*

Proof. Assume that $\forall r$, $\lambda_r > 0$. In this case, Eq. 4.39 implies that $\forall r$, $\hat{\delta}_r = 0$. This implies that $\sum_{r=1}^{K} \hat{\delta}_r$ is equal to 0, and the delay budget, $\hat{\Delta}$, remains totally unused. This violates the

optimality property of the solution. Therefore, there exists at least one Lagrange multiplier λ_r that is strictly equal to 0. □

Lemma 5. *If Opt_LD violates some inequality constraints given by Eq. 4.34 and Eq. 4.35, then* $\exists r$ *such that* $\lambda_{2k+r} > 0$ *and* $\lambda_{3K+r} > 0$.

Proof. Assume $\exists r$ such that $\lambda_{2k+r} > 0$ and $\lambda_{3K+r} > 0$. In this case, $\sigma_r = \sigma_r^{min}$ and $\sigma_r = \sigma_r^{max}$ at the same time, which results in the occurrence of inequality constraint violations in Eq. 4.34 and Eq. 4.35. Therefore, $\forall r$, the value of $\lambda_{2k+r} \cdot \lambda_{3K+r}$ is strictly equal to 0, which implies $\forall r$, such that (i) $\lambda_{2k+r} > 0$ and $\lambda_{3K+r} = 0$, or (ii) $\lambda_{2k+r} = 0$ and $\lambda_{3K+r} > 0$, or (iii) $\lambda_{2k+r} = 0$ and $\lambda_{3K+r} = 0$. □

4.5.6.3 *Opt_EDFS* Solution *Opt_EDFS* is characterized by the set $Y = \{y_r() | y_r(\sigma_r, \hat{\delta}_r) = (a_r \cdot (l_r + \hat{\delta}_r) + b_r) \cdot (\sigma_r)^2\}$ the set $\hat{H} = \left\{\hat{h}_1, \cdots, \hat{h}_K\right\}$ of upper bounds, and the end-to-end delay budget, $\hat{\Delta}$. The optimization problem can be expressed as

$$\textbf{minimize} \quad \mathcal{E}_p^D(\vec{\sigma}, \vec{\hat{\delta}}) = \sum_{r=1}^{K} (a_r \cdot (l_r + \hat{\delta}_r) + b_r) \cdot (\sigma_r)^2 \tag{4.47}$$

$$\textbf{subject to} \quad 0 - \hat{\delta}_r \leq 0, \ (1 \leq r \leq K) \tag{4.48}$$

$$\hat{\delta}_r - \hat{h}_r \leq 0, \ (1 \leq r \leq K) \tag{4.49}$$

$$\sigma_r^{min} - \sigma_r \leq 0, \ (1 \leq r \leq K) \tag{4.50}$$

$$\sigma_r - \sigma_r^{max} \leq 0, \ (1 \leq r \leq K) \tag{4.51}$$

$$\sum_{r=1}^{K} \hat{\delta}_r - \hat{\Delta} = 0 \tag{4.52}$$

Furthermore, we have $0 < \hat{\Delta} < \sum_{r=1}^{K} \hat{h}_r$ and $0 < \hat{h}_r, \forall i$. *Opt_LD* differs from *Opt_EDFS* in the additional set of upper bound constraints of delays. Consequently, it is easy to show

that if S_{Opt_LD} satisfies the constraints of $\hat{\delta}_r - \hat{h}_r \leq 0, r = 1, \cdots, K$, the set S_{Opt_LD} is a feasible solution for Opt_EDFS, and $S_{Opt\_L} = S_{Opt\_EDFS}$. However, if an upper bound constraint is violated, an iterative process, in a way analogous to the process used to derive S_{Opt_LD}, must be used to remove upper bound constraint violations.

Let $U = \{m | -y_m^{'}(\sigma_m, \hat{\delta}_m) \geq -y_r^{'}(\sigma_r, \hat{\delta}_r), \forall r\}$. The set U contains the functions $y_m \in Y$, such that $-y_m^{'}()$ leads to the largest marginal returns at the upper bounds.

Algorithm 4.1 $Opt\_EDFS(Y, \hat{H}, \hat{\Delta})$.

1: **Set** $S_{Opt\_EDFS} = \oslash$
2: **if** $Y = \oslash$ **then**
3: **exit**
4: **end if**
5: **Find** S_{Opt_LD} **by invoking algorithm** Opt_LD
6: **if** *all upper bounds are satisfied* **then**
7: **if** *computed execution rates are feasible* **then**
8: $S_{Opt\_EDFS} = S_{Opt\_EDFS} \cup S_{Opt\_LD}$
9: **exit**
10: **else**
11: **Compute** U
12: **end if**
13: **end if**
14: **Set** $\hat{\delta}_q = \hat{h}_q$, $\forall q \in U$ **in** S_{Opt_EDFS}
15: **Set** $\hat{\Delta} = \hat{\Delta} - \sum_{m \in U} \hat{h}_m$
16: **Set** $Y = Y - U$
17: **Set** $\hat{H} = \hat{H} - \left\{ \hat{h}_k | k \in U \right\}$
18: **Go to step 2**

The algorithm $Opt\_EDFS()$, depicted in Algorithm 4.1, solves the Opt_EDFS problem based on successive invocations of Opt_LD. First, we find the solution of the corresponding Opt_LD problem. The solution, if it exists, is optimal for the Opt_LD problem, which does not take into account upper bound constraints. If the upper bound constraints are automatically satisfied, S_{Opt_LD} is also optimal for the Opt_EDFS problem. However, if this is not the case, the correctness of the algorithm needs to be further argued in a similar fashion as in the case of the Opt_LD problem. Deriving the necessary and sufficient Kuhn-Tucker

conditions for problem Opt_EDFS after considering the upper bounds, results in

$$a_r \cdot (\sigma_r)^2 + \upsilon - \lambda_r + \lambda_{K+r} = 0,\ r = 1, \cdots, K \tag{4.53}$$

$$\begin{aligned} &\lambda_r \cdot \frac{\partial l_r(\sigma_r)}{\partial \sigma_r} + 2 \cdot \sigma_r \cdot (a_r \cdot (l_r(\sigma_r) + \hat{\delta}_r) + b_r) - Z_r = 0, \\ &where\ Z_r = \lambda_{2K+r} - \lambda_{3K+r},\ r = 1, \cdots, K \end{aligned} \tag{4.54}$$

$$-\lambda_r \cdot \hat{\delta}_r = 0,\ r = 1, \cdots, K \tag{4.55}$$

$$\lambda_{K+r} \cdot (\hat{\delta}_r - \hat{h}_r) = 0,\ r = 1, \cdots, K \tag{4.56}$$

$$\lambda_{2K+r} \cdot (\sigma_r^{min} - \sigma_r) = 0,\ r = 1, \cdots, K \tag{4.57}$$

$$\lambda_{3K+r} \cdot (\sigma_r - \sigma_r^{max}) = 0,\ r = 1, \cdots, K \tag{4.58}$$

$$\lambda_r, \lambda_{K+r}, \lambda_{2K+r}, \lambda_{3K+r} \geq 0,\ r = 1, \cdots, K \tag{4.59}$$

where $\lambda_j, j = 1, \cdots, 4K$ and υ are the Lagrange multipliers. Using the fact that $\sum_{r=1}^{K} \hat{\delta}_r = \hat{\Delta}$, $0 \leq \hat{\delta}_r \leq \hat{h}_r$, and Eq. 4.53$\sim$4.59, results in

$$\sigma_r = \sqrt{\frac{|\lambda_r - \lambda_{K+r} - \upsilon|}{a_r}},\ r = 1, \cdots, K \tag{4.60}$$

$$\begin{aligned} \delta_r = &\frac{\lambda_{2k+r} - \lambda_{3K+r} - \lambda_r \cdot \frac{\partial l_r(\sigma_r)}{\partial \sigma_r}\Big|_{\sigma_r = \sqrt{\frac{|\lambda_r - \lambda_{K+r} - \upsilon|}{a_r}}}}{2\sqrt{a_r \cdot |\lambda_r - \lambda_{K+r} - \upsilon|}} - \frac{b_r}{a_r}, \\ &r = 1, \cdots, K \\ \hat{\delta}_r = &\delta_r - l_r(\sigma_r = \sqrt{\frac{|\lambda_r - \lambda_{K+r} - \upsilon|}{a_r}}),\ r = 1, \cdots, K \end{aligned} \tag{4.61}$$

It can easily be shown that if S_{Opt_LD} violates upper bound constraints given by Eq. 4.49 then $\exists i,\ \lambda_{K+r} > 0$. A similar argument, which states that if $\exists i,\ \lambda_{K+r} > 0$ then $\lambda_r = 0$, can also be proven. Besides, it also can be proved that $\forall i$, such that $\lambda_{2K+r}, \lambda_{3K+r} \geq 0$ and

$\lambda_{2K+r} \cdot \lambda_{3K+r} = 0$, which implies $\forall r$, such that (i) $\lambda_{2k+r} > 0$ and $\lambda_{3K+r} = 0$, or (ii) $\lambda_{2k+r} = 0$ and $\lambda_{3K+r} > 0$, or (iii) $\lambda_{2k+r} = 0$ and $\lambda_{3K+r} = 0$. These observations can then be used to prove that if the Lagrange multipliers $\lambda_{K+r} > 0$, $r \in U$, are all nonzero, this implies, based on Eq. 4.56, that $\hat{\lambda}_q = \hat{h}_q$, $\forall q \in U$.

4.5.7 Delay Assignment Heuristics

One possible approach to compute suitable per-router delay values considers the processing capabilities of the node as the limiting factor, which is used to estimate the per-router delay values for the new flow along with its end-to-end delay requirement, Δ. This policy is likely to achieve more efficient use of the network resources, which in turn increases the capability of a node to support future flow requests. A different approach would be to balance the load across the routing path in order to minimize the likelihood bottlenecks. To achieve this goal, the policy assigns larger per-router delay budgets to highly loaded nodes than to lightly loaded nodes. This is based on the observation that assigning a large delay value to a given flow causes a relatively small load increase to a heavily loaded node. In the following, we describe two heuristics, namely processing-capacity based heuristic, $PCH()$, and load balancing heuristic, $LBH()$. $PCH()$ aims at reducing the processing load placed by the new flow over the node, while $LBH()$ attempts to distribute the load uniformly across the routing path. The performance of these two heuristics is then compared to the optimal policy.

4.5.7.1 Processing-capability based heuristic, $PCH()$

The basic steps of the processing-capability based heuristic, $PCH()$, are described in Algorithm 4.2. The input parameters of $PCH()$ include the delay bounds, l_r and h_r, for each router along the routing path, derived from the new flow traffic rate specification and the end-to-end delay requirement, Δ, of the underlying application.

The approach used by $PCH()$ is to compute a potential delay value, Δ_r, which only considers per-router processing capabilities. This is achieved by taking the routing paths current delay and distributing it proportionally across all nodes on the routing path. There-

fore, Δ_r is set to be equal to $\Delta \cdot \frac{l_r}{\sum_{r \in p} l_r}$. Based on processing-capability based Δ_r, $PCH()$ uses Exponentially Weighted Moving Average (EWMA) algorithm to predict the per-router delay upper bounds, $\Delta_r^P, (1 \leq r \leq K)$, for the new flow.

$$\Delta_r^P = (1 - \pi_{PCH}) \cdot \Delta_r^{P\_old} + \pi_{PCH} \cdot \cdot \Delta_r \tag{4.62}$$

where the smooth factor, $\pi_{PCH} \in [0, 1]$, is used to guarantee that the predicted delay value is not affected by small deviations. Notice that the sum over of Δ_r^Ps does not exceed the new flows end-to-end delay requirement, Δ. To address this limitation, if it occurs, Δ_r^Ps are adjusted to h_rs, which are defined in Section 4.5.2.2.

Algorithm 4.2 $PCH()$.

1: **Initialize** $\Delta_r^{P\_old}$**s to be** h_r**,** $r \in \{1, ..., K\}$
2: **for** $r \in \{1, ..., K\}$ **do**
3: $\quad \Delta_r = \Delta \cdot \frac{l_r}{\sum_{r \in p} l_r}$
4: $\quad$ **Predict the per-delay upper bound values for the new flow**
5: $\quad \Delta_r^P = (1 - \pi_{PCH}) \cdot \Delta_r^{P\_old} + \pi_{PCH} \cdot \cdot \Delta_r$
6: **end for**
7: **if** $\sum_{i=1}^{K} \Delta_r^P > \Delta$ **then**
8: $\quad$ **for** $r \in \{1, ..., K\}$ **do**
9: $\quad\quad \Delta_r^P = h_r$
10: $\quad$ **end for**
11: **end if**
12: $\Delta_r^{P\_old} = \Delta_r^P$

4.5.7.2 Load-balancing based heuristic, $LBH()$ Different from to $PCH()$, load balancing heuristic, $LBH()$, attempts to balance the load along the routing path when accepting a flow request. It does so by computing the initial lower bound delay values, Δ_rs, to be proportional to the nodes' respective loads and then adjusting these values such that they lie within each node's smallest and largest feasible delay values, without violating the end-to-end delay requirement of the flow. Therefore, Δ_r is set to be equal to $\Delta \cdot \frac{\vartheta_r}{\sum_{r \in p} \vartheta_r}$, where ϑ_r denotes the current load at node r. Based on this strategy, a lightly loaded node is assigned a smaller delay value and thus takes on a higher load, while a highly loaded node is assigned

a higher delay value and sees a smaller increase in its load. $LBH()$ attempts to distribute the load uniformly across the routing path.

Similarly, based on processing-capability based Δ_r, $LBH()$ also uses the EWMA algorithm to predict the per-router delay upper bounds, $\Delta_r^P, (1 \leq r \leq K)$, for the new flow.

$$\Delta_r^P = (1 - \pi_{LBH}) \cdot \Delta_r^{P\_old} + \pi_{LBH} \cdot \cdot \Delta_r \tag{4.63}$$

where the smooth factor, $\pi_{LBH} \in [0, 1]$, is used to guarantee that the load-balancing based predicted delay value is not affected by small deviations.

In addition to the smallest and largest delay values, l_r and h_r, supported by each node $i : 1, \cdots, K$ along the routing path and the maximum end-to-end delay requirement, Δ, of the new flow, $LBH()$ includes the current load, ϑ_r, at node r as input parameter. The heuristic uses the information about the current workloads of the routers to achieve a balanced delay assignment across the routing path. The basic steps of the heuristic are described in Algorithm 4.3. Initially, $LBH()$ computes a delay value $\Delta_r = \Delta \cdot \frac{\vartheta_r}{\sum_{r \in p} \vartheta_r}$ to be proportional to the load of each node along the routing path. Notice that the sum of Δ_r^Ps does not exceed the new flows end-to-end delay requirement, Δ. To address this limitation, if it occurs, Δ_r^Ps are adjusted to h_rs. $LBH()$ attempts to adjust this value to meet the delay constraints of each node along the routing path discussed above. The procedure used by $LBH()$ to adjust the initial lower bound delay value, l_r, assigned to each node. A predicted upper bound delay value Δ_r^P is computed based exclusively on the current traffic loads of the routing routers. If the minimum within the interval $\left[l_r, \Delta_r^P\right]$ is not feasible, a search procedure to locate a feasible value within the interval $[l_r, h_r]$ is initiated. The search continues until a feasible value is determined.

In the following section, we present a simulation framework used to evaluate the performance of the proposed energy- and delay-aware flow scheduling algorithm and its two heuristics.

Algorithm 4.3 $LBH()$.

1: **Initialize** $\Delta_r^{P\text{-}old}$**s to be** h_r**,** $r \in \{1, ..., K\}$
2: **for** $r \in \{1, \cdots, K\}$ **do**
3: $\quad \Delta_r = \Delta \cdot \frac{\vartheta_r}{\sum_{r \in p} \vartheta_r}$
4: $\quad$ **Predict the per-delay upper bound values for the new flow**
5: $\quad \Delta_r^P = (1 - \pi_{LBH}) \cdot \Delta_r^{P\text{-}old} + \pi_{LBH} \cdot \cdot \Delta_r$
6: **end for**
7: **if** $\sum_{i=1}^{K} \Delta_r^P > \Delta$ **then**
8: $\quad$ **for** $r \in \{1, ..., K\}$ **do**
9: $\quad\quad \Delta_r^P = h_r$
10: $\quad$ **end for**
11: **end if**
12: $\Delta_r^{P\text{-}old} = \Delta_r^P$

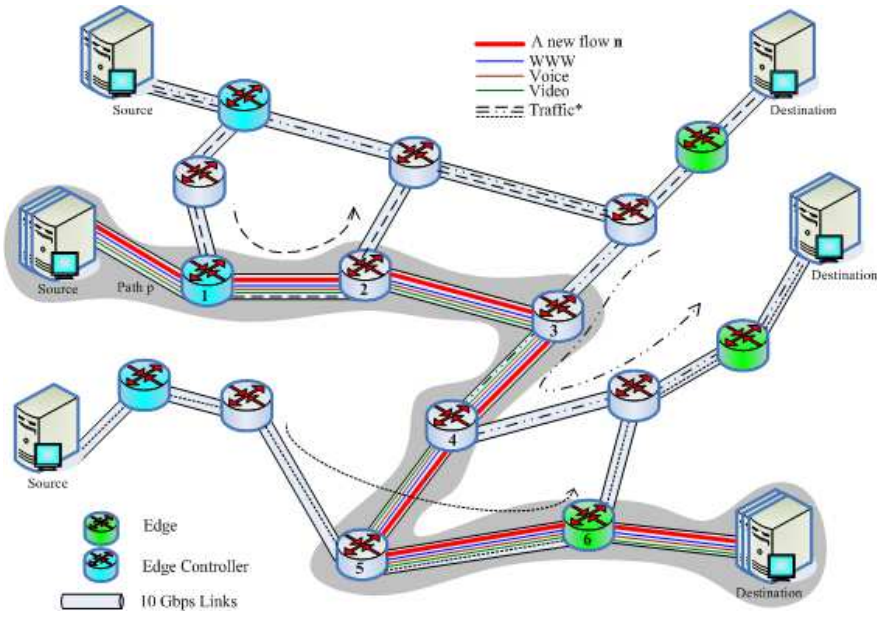

Figure 4.1: A new connection request along a path, p, in a network topology, N.

Table 4.1: Traffic source models and specifications.

	Traffic Class	**Traffic Model**	**Average Packet Length (bytes)**	**QoS**	**LBAP Parameters**	
				End-to-end Delay (ms)	**Average Burst** (β **kbits**)	**Average Rate** (ρ **kbps**)
WWW	Interactive	Exponential	1250	≤ 150	≤ 40	≤ 2000
Voice	Interactive	Exponential	1250	≤ 150	≤ 62.67	≤ 64
Video	Streaming	Exponential	1250	≤ 150	≤ 320	≤ 1660
Comb	WWW:Voice:Video	Exponential	1250	≤ 150	refer above	refer above
Traffic*	WWW	Exponential	1250	≤ 150	$3.0 \sim 7.0$	$42.4 \sim 122.4$

Table 4.2: Main simulation parameters and conditions.

Items	Simulation Parameters	Simulation Values
Router	NIC port	$10GE$
	$CPI(cycles/instruction)$	1.2
	$\psi^{min}(GHz)$	$0.6 \sim 1.6$
	$\psi^{max}(GHz)$	2.4
	$\left[\psi^{min}, \psi^{max}\right](GHz)$	1) $[1.6, 2.4]$ [Intel, 2012]
		2) $[0.6 \sim 1.6, 2.4]$
Packet	Packet Max size ($bytes$)	1500
	$IPB(instructions/byte)$	1.6
Network	Propagation Delay (ms)	30
	Traffic Model	WWW,Voice,Video
		Comb (WWW:Voice:Video)
Others	Random Generator ($Seed$)	0,1,2

Table 4.3: Energy-efficient Metrics.

Metrics	Definition	Description
DEG(%)	$\frac{\mathcal{E}_p^D(\vec{\sigma}^{max},\vec{\delta})-\mathcal{E}_p^D(\vec{\sigma},\vec{\delta})}{\mathcal{E}_p^D(\vec{\sigma}^{max},\vec{\delta})}$	Dynamic Energy Gain (static energy is not included)
PG (%)	$\frac{\varphi(\vec{\sigma}^{max})-\varphi^D(\vec{\sigma})}{\varphi(\vec{\sigma}^{max})}$	Power Gain (both static and dynamic power are included)
DPR (%)	$\frac{\Delta_r^P-h_r}{h_r}$	Delay Prediction Ratio (for delay bound assignment)
DEGR (%)	$\frac{DEG_{Heuristic}-DEG_{EDFS}}{DEG_{EDFS}}$	Dynamic Energy Gain Ratio (static energy is not included)

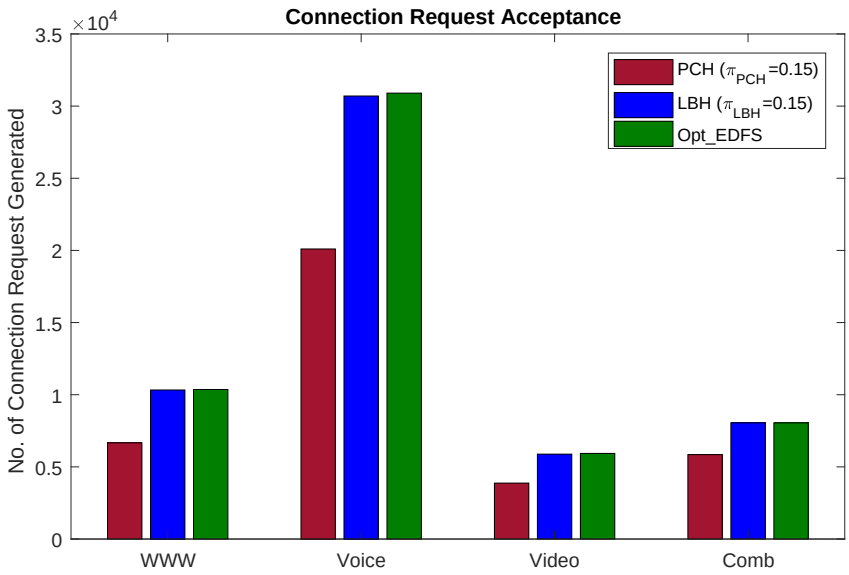

Figure 4.2: Connection request acceptance.

4.6 Performance Evaluation

In order to assess the efficiency of the proposed energy- and delay-aware flow management strategy and its two heuristics, we developed a simulation framework to carry out a set of simulation-based experiments. The network topology in this study is depicted in Figure 4.1. The focus of the analysis is on the shaded path, p, carrying QoS flows from source to destination. All line cards (LCs) in each router are configured with multiple network processor units (NPUs); each unit uses a DVFS-enabled, delay-based scheduling policy. We set the capacity of all network links to 10 *Gbps*. Two cases of processor execution rates are considered. In the first case, the frequency range is $[1.6, 2.4]$ GHz, which is used in Intel XEON DPDK line card [Intel, 2012]. In the second case, the minimum frequency, ψ^{min}, of a router is randomly selected from the range [0.6, 1.6] GHz and the maximum frequency, ψ^{max} is set to 2.4 GHz.

Five classes of traffic are simulated. These classes and their traffic rate specifications and QoS requirements are listed in Table 4.1. The ITU G.114 specification recommends less

than 150 ms one-way end-to-end delay for high-quality real-time traffic. Consequently, the end-to-end delay requirements of the traffic flows are randomly generated from an interval, $[0, 150]$ ms. The number of flows, within each traffic class, is varied to generate different network loads.

Table 4.2 describes the main simulation parameters used in this simulation study. In addition, two different metrics are defined, namely *dynamic energy gain* (DEG) and *power gain* (PG). The first metric is to measure the relative dynamic energy gain of a processor running at an execution rate, σ over a processor running at a maximum execution rate, σ^{max}. The second measures the relative power gain. These metrics are defined in Table 4.3. Besides, another two ratios, namely *delay prediction ratio* (DPR) and *dynamic energy gain ratio* (DEGR), are provided in Table 4.3, which are two energy-efficient metrics as auxiliary use to help comparison between the optimal algorithm Opt_EDFS and its two heuristics.

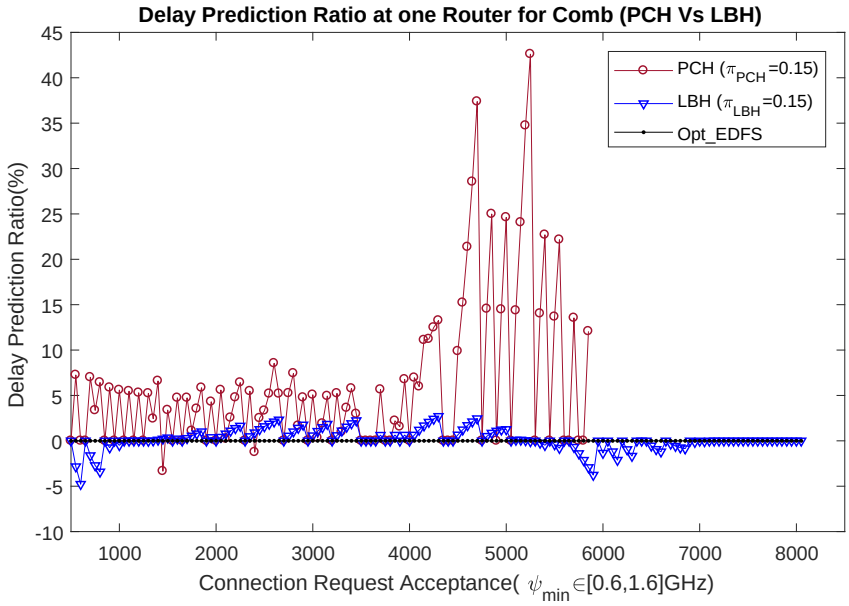

Figure 4.3: DPR comparison between PCH and LBH.

4.6.1 Comparison with Two Heuristics

In this chapter, two delay assignment heuristics, namely $PCH()$ and $LBH()$ are proposed to compared with the optimal algorithm Opt_EDFS. we carried a series of experiments to

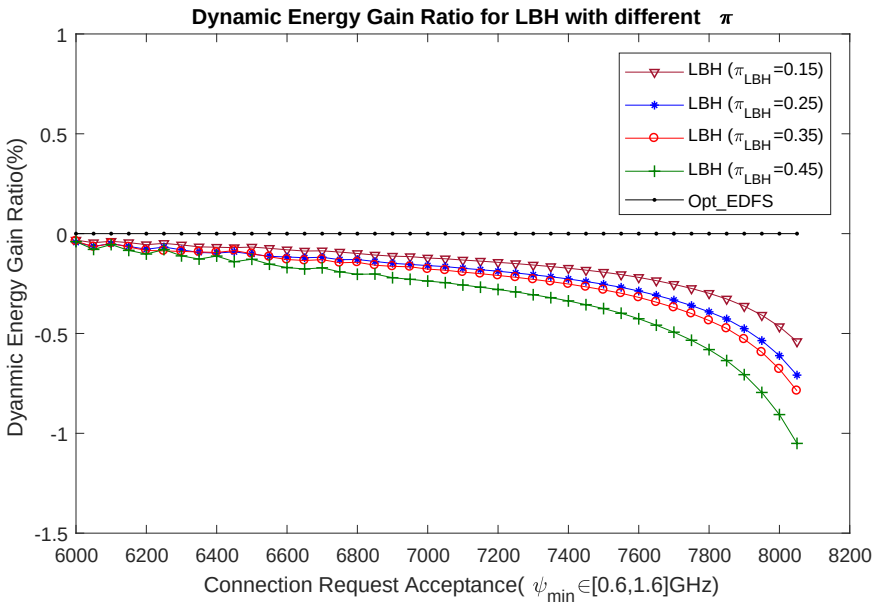

Figure 4.4: DEGR for LBH with different π.

explore the energy saving compared with the optimal algorithm Opt_EDFS to different parameters. This set of experiments is to assess the performance of different classes of traffic, based on two metrics: DPR and DEGR described in Table 4.3. The simulated traffic is generated by web, voice and video flows. The traffic and QoS specification parameters of these applications are described in Table 4.1, whereby background traffic is used to vary the network load.

Fig.4.2 depicts the number of accepted flows for different classes of traffic, namely WWW, Voice, Video, and Comb (WWW, Voice, and Video are combined by 1:1:1). It shows that the connection request acceptance of $LBH()$ is approximate to Opt_EDFS, while the acceptance of $PCH()$ is much lower than $LBH()$ due to its high sensitivity to delay prediction based on router processing capacities, as displayed in Figure 4.3. In other words, $PCH()$ could not find feasible solutions any more for its unacceptable requests which, however, still are in the range of the acceptance of Opt_EDFS and $LBH()$.

Contrary to $PCH()$, $LBH()$ shows its more flexibility to estimate delay bound values to adapt to the current traffic load. A series of values of π in range $[0.1; 0.5]$ is tested

for $LBH()$. Fig.4.4 depicts a series of dynamic energy gain ratios (DEGRs) of LBH with different $\pi = 0.15, 0.25, 0.35, 0.45$ for traffic Comb. The results show that energy saving difference between Opt_EDFS and $LBH()$ is only within $\pm 1.5\%$, whereby Opt_EDFS, especially $\pi = 0.45$, has slight larger energy saving than $LBH()$ for higher traffic load along the routing path with K routers, which have different minimum frequency values, $\psi_r^{min} \in [0.6, 1.6]\, GHz (1 \leq r \leq K)$. Therefore, Opt_EDFS is our focus for the rest of experiments.

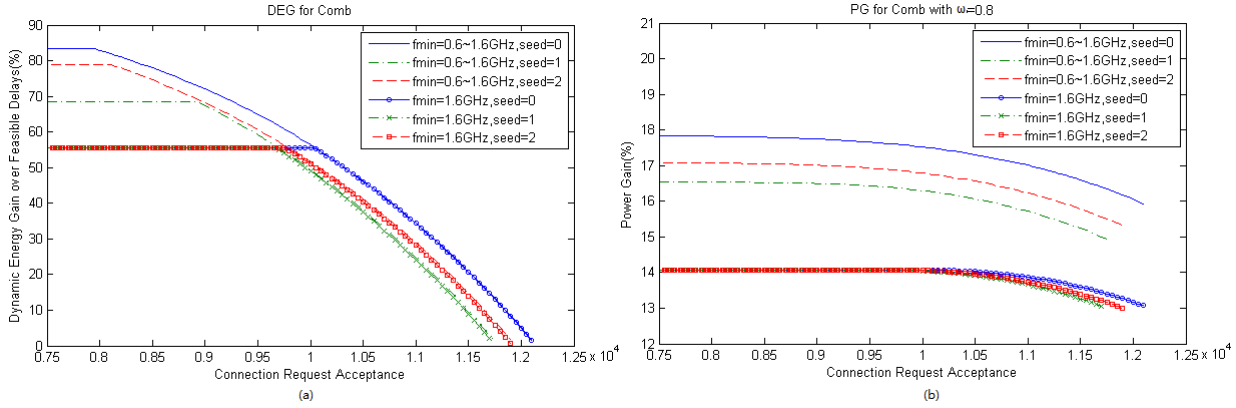

Figure 4.5: (a) Dynamic energy gains and (b) Power gains for the combination traffic source under fixed $\psi^{min} = 1.6\ GHz$ and randomly generated $\psi^{min} \in [0.6, 1.6]\ GHz$.

4.6.2 Energy and Power Gain Evaluation of Opt_EDFS

The objective of this set of experiments is to assess the performance of different classes of traffic described in Table 4.1 through the same optimal algorithm Opt_EDFS, in terms of the energy and power gains metrics described in Table 4.3.

In these experiments different minimum frequency values, ψ^{min} are used, where each router, along the path, is randomly assigned a minimum frequency from the interval $[0.6, 1.6]\ GHz$. Fig.4.5 shows the dynamic energy gain (DEG) and power gain (PG) for the simulated traffic. The results also show that the variability of minimum frequency among the routers leads to higher DEGs and PGs. The results show that the highest DEG and PG gains are achieved when the router's minimum frequency, $\psi_r^{min} (1 \leq r \leq K)$, varies along the path. In a homogeneous network, where the router's minimum frequency is fixed, the DEG and PG gains

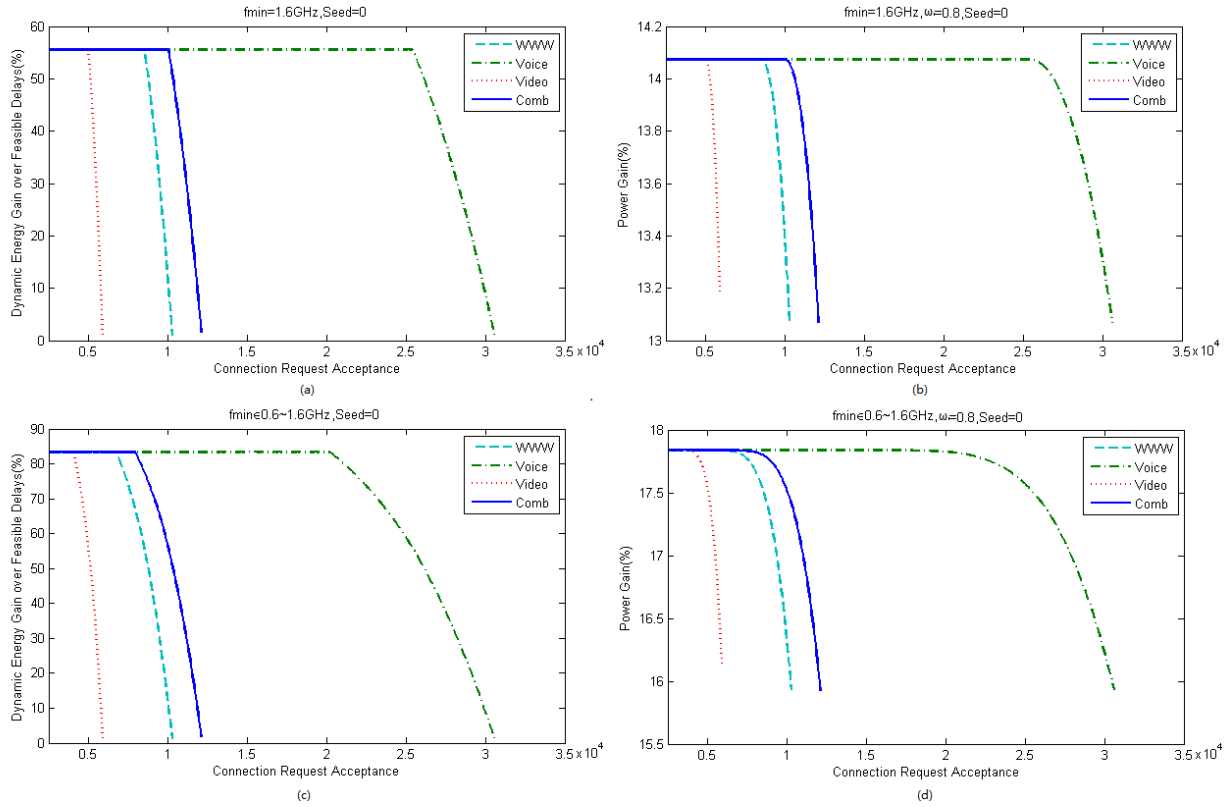

Figure 4.6: Dynamic energy gains and power gains for different traffic sources under (a,b) fixed $\psi^{min} = 1.6\ GHz$ and (c,d) randomly generated $\psi^{min} \in [0.6, 1.6]\ GHz$.

are lower, than those obtained in a heterogeneous network. The number of accepted flows for each class of traffic is depicted as Fig.4.2.

Fig.4.6(a) depicts the DEG as a function of the number of flow requests accepted by the network, for a fixed minimum frequency across all routers. The minimum frequency for a router is set to $\psi_r^{min} = 1.6\ GHz\ (1 \leq r \leq K)$. In this case, the DEGs are up to 55.56%. The results show higher DEGs are achieved when the traffic load is low. In the second experiment, the range of router's minimum frequency, $\psi_r^{min}(1 \leq r \leq K)$, is set to $[0.6, 1.6]\ GHz\ (1 \leq r \leq K)$. The results, depicted in Figure 4.6(c), show that when the routers' minimum frequency varied along the routing path, the DEG gains can be as high as 83.33%. The results also show that the DEGs decrease as the network load increases along the given path.

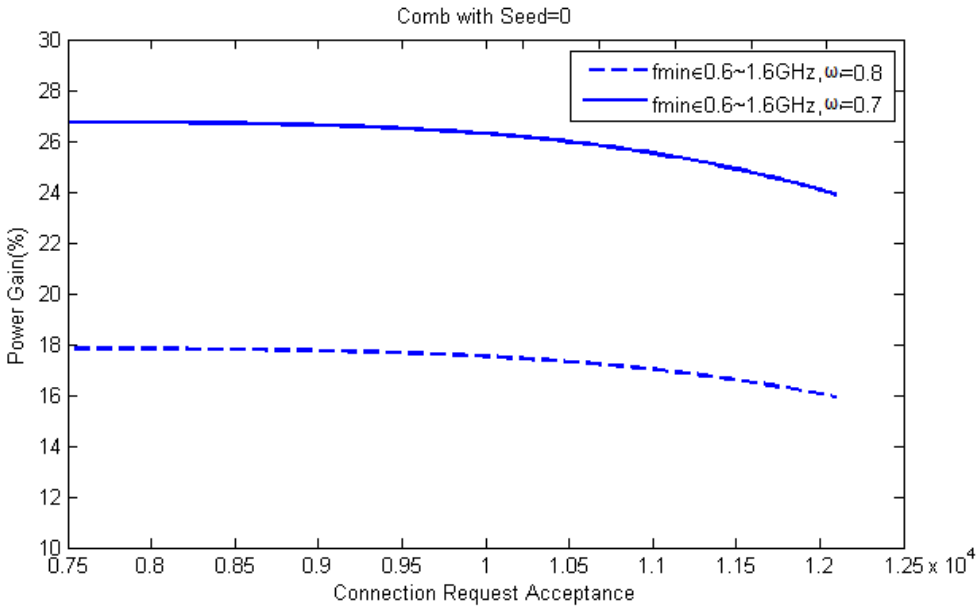

Figure 4.7: Power gains comparisons under different values of the static power ratio ω.

Similar behavior is observed with respect to PG gains. The results depicted in Figure 4.6(b) and Fig.4.6(d) show that PGs are achieved for all traffic classes. Furthermore, the results show that as the number of accepted flows increases, the power saving decreases. It is worth noting that higher PGs are achieved, when the value of the static power ratio, ω, decreases. A decrease in ω results in an increase in the proportion of dynamic power.

The results in Figure 4.7 show when the minimum frequency, ψ_r^{min}, is in the range $[0.6, 1.6]$ GHz, the maximum frequency is ψ_r^{max}, is set to 2.4 GHz $(1 \leq r \leq K)$ and ω is set to 0.8, up to 17.84% PG can be achieved. When ω is set to 0.7, the PG can reach up to 26.76%.

4.7 Conclusions

The focus on this chapter is the development of an energy- and delay-aware flow control and management framework to support the QoS requirements of delay-sensitive applica-

tions, while minimizing network energy consumption. Based on the flow traffic and QoS specification, per-router delay budget and per-router execution rate are computed, so that energy consumption is minimized without violating the flow's *end-to-end* delay requirement. A model to compute a path-based energy consumption, taking into consideration both the static and dynamic energy components, is developed, and a methodology to compute feasible lower and upper bound delays of given flow, based on the router's current traffic load, is proposed. A simulation framework assesses the performance of the proposed strategy and its two heuristics in terms of different energy-efficient metrics. The simulation results show that the proposed delay-aware strategy, $EDFS$, achieves optimal energy and power saving, without violating the QoS requirements of the underlying applications. The achieved gains are higher when the network load is low. The results demonstrate that up to 83.33% dynamic energy saving of total dynamic energy consumption and up to 26.76% power saving of total power consumption can be achieved.

Generally, for optimal results, scheduling and speed scaling should be considered simultaneously. In many cases, the optimal combination of scheduling and speed scaling is NP-hard. This follows from the fact that multiprocessor scheduling, which is already NP-hard, is a special case of the generally combined speed scaling and scheduling problem of delay-sensitive traffic load. Moreover, the energy saving through scaling speed is limited due to the load-dependent power source percentage occupancy among the whole power consumption, that is to say, the dynamic energy saving depends on the changes in the traffic load so that the proposed DVFS-based techniques only can solve the problem of dynamic energy saving in network components. However, another problem emerges that the energy consumption of current IP networks is not proportional to the utilization level. Even in low or no usage context, network equipment consumes energy at a high level. Therefore, compared to slow-down approaches to save dynamical energy, shutdown approaches could achieve the goal of saving more energy by shutting down network equipment or its components (i.e. put them into sleep modes) when they are idle or low-demand. Therefore, seeking an intelligent sleep-based power- or energy-aware strategy to save more power/energy consumption without QoS violation becomes our next goal.

5.0 Sleep-based Power Management and a Traffic-aware Strategy

The power consumption of current network devices is not always proportional to their utilization. Regardless of the traffic level, the network is constantly operating near maximum power. Furthermore, many related studies have shown that the base system (including chassis, switch fabric, and router processor) of a network device is the major contributor to its overall energy usage. Shutting down the routers, therefore, could save more energy than slowing down the processor unit components such as the line cards. The latter strategies are explored in Chapter 3 and Chapter 4, using DVFS-based, energy- and QoS-aware power management. In this chapter, we investigate a new strategy to save the network power consumption, focusing on sleep-based, traffic-aware power management. This proposed strategy aims at adapting the whole network power consumption to the traffic levels by reconfiguring the network and putting to sleep the lightly loaded network elements, while taking into account network performance requirements. Moreover, sleep mode technique is considered jointly with speed scaling technique to further tune power and energy consumption, while adhering close to QoS requirements. The results show that applying this strategy can lead to power savings of up to 62.58% of the total power consumption.

5.1 Introduction

Compared to speed scaling, switching off routers, when the network traffic is low, holds great promise to achieve higher energy saving [Gupta and Singh, 2003]. As such, incorporating power control and energy-awareness into network management has become a critical objective in the design of future networks, as it provides a viable solution to minimize power and reduce energy consumption [Pierson, 2015]. The challenge is to develop an efficient strategy that can dynamically adapt to the network traffic load to strike a balance between minimizing power and energy consumption and adhering to QoS requirements of the network supported applications. This challenge stems from the fact that switching off a router too

soon may lead to severe QoS degradation, if the traffic abruptly increases immediately after a decision to move the router into sleep mode is made. This requires new insights on how network traffic is dynamically and accurately predicted. Furthermore, scalable network control frameworks that effectively integrate power control and traffic management to minimize energy consumption, while adhering to the QoS requirements of the underlying applications, must be investigated [Addis et al., 2016].

To address this shortcoming, we propose a framework to explore the design and assess the performance of a sleep-based, traffic-aware power management strategy, referred as to STAPM. STAPM dynamically adapts network power consumption to network load and uses agile network configuration to reroute traffic flows around switched off routers toward their destinations, without severally degrading the network performance. To further achieve higher levels of energy consumption, the proposed strategy will be seamlessly integrated with a speed scaling based approach. Combined, the two approaches hold great potential to achieve significant power saving, while supporting the QoS requirements of the underlying applications. The main contributions of this chapter are: (i) the development of a model to compute a network-based power consumption, taking into consideration both the static and dynamic power components, (ii) the design of a traffic-aware power management strategy to switch off lightly loaded network elements, based on the router's current traffic load and network congestion, and (iii) the development of agile network reconfiguration techniques to ensure close adherence to applications' QoS requirements. A simulation framework is developed to assess the performance of the proposed strategy, focusing on three power-efficient green metrics, namely power gain (PG), dynamic power gain (DPG), and static power gain (SPG).

The rest of this chapter is organized by sections as follows: the related work is reviewed in Section 5.2. The sleep-based, traffic-aware power controller and its architecture design are introduced in Section 5.3. Within this framework, a sleep-based, traffic-aware power control and management strategy is discussed. The performance of the proposed strategy is assessed in Section 5.4. Finally, Section 5.5 presents the conclusion of this chapter.

5.2 Related Work

Device sleeping represents a viable solution to energy saving because the consumption of current network devices is not proportional to the utilization level. As such, when routers are constantly active, the overall network consumption remains close to maximum power consumption. As introduced in Chapter 2, since the consumption of the base system is the major contributor to the overall network power consumption, shutting down the router, therefore, could save more energy than only switching off its line cards. Despite its benefits, switching off routers raise several network challenges, including rerouting potentially a very large number of connections around switched off routers toward their destinations, the likelihood of extended wake-up periods that leed to high levels of traffic congestion, especially in bursty network environments. Furthermore, the energy cost of switching off and on routers may become prohibitive if these operations are undertaken frequently and the network traffic is highly variable. Consequently, the decision to switch off a router to save energy must be carefully weighted against the energy needed to reactive a sleeping router. An important factor that impacts such a decision is the likelihood of traffic increase in the immediate future.

Some researches choose to switch off individual line cards and remap the links to other ones. This avoids discontinuities and saves power when the traffic load is light. Fisher et al. propose a form of infrastructure sleeping where they shut down the lightly loaded cables and the line cards, instead of the whole router, during periods of low utilization in [Fisher et al., 2010]. In [Idzikowski et al., 2010], routing reconfiguration at a different layer, namely IP layer (the virtual layer) and WDM (the physical layer), for achieving energy saving through switching off line cards, is compared. The scheme that rerouters demands in the virtual layer achieves the best energy saving. Similarly, Shang, et al. also propose a scheme to switch off line cards when traffic load is low in [Zhang et al., 2010b]. In order to improve the energy efficiency of backbone networks by dynamically adjusting the number of active links according to network load, Carpa et al. propose an intra-domain software-defined network (SDN) approach in [Carpa et al., 2015], an energy-aware traffic engineering technique, to select and turn off a subset of links. The implemented solution shows that as

much as 44% of links can be switched off to save energy in real backbone networks. Recently, Virtualized Network Environment (VNE) has recently emerged as a solution to address the challenges of the future Internet. It is essential to develop novel techniques to reduce VNEs energy consumption. Ghazisaeedi et al. propose a novel optimization algorithm for VNE in [Ghazisaeedi et al., 2012], by sleeping reconfiguration on the maximum number of physical links during off-peak hours, while still guaranteeing the connectivity and off-peak bandwidth availability for supporting parallel virtual networks over the top. Simulation results based on the GANT network topology show this algorithm is able to put a notable number of physical links to sleep during off-peak hours while still satisfying the bandwidth demands requested by ongoing traffic sessions in the virtual networks. It, however, does not change the mapping of VNs, this decreases the level of energy saving. The same authors propose an energy saving method that optimizes VNEs energy consumption during the off-peak time in [Ghazisaeedi and Huang, 2015]. This method reconfigures mapping for some of the embedded virtual links in the off-peak period. The proposed strategy enables providers to adjust the level of the reconfiguration, and accordingly control probable traffic disruptions due to the reconfiguration. This problem is formulated as a Binary Integer Linear Program (BILP). the defined BILP is NP-hard, a novel heuristic algorithm is also suggested. The proposed energy saving methods are evaluated over random VNE scenarios. The results confirm the defined solutions are able to save notable amounts of energy during off-peak period, while still accommodating off-peak traffic demands of involved virtual networks.

Other researches, however, consider to put devices and their components into sleep mode when they are not used, to save more energy. Gianoli and Giovanni propose an energy-aware traffic engineering solution to minimize the energy consumption of the network through a management strategy that selectively switches off devices according to the traffic level, and model a set of traffic scenarios corresponding to different time periods and consider a set of traffic scenarios and jointly optimize their energy consumption assuming a per-flow routing in [Gianoli, 2014]. Chiaraviglio et al. propose an algorithm that can selectively turn off some nodes and links of an IP-based backbone network during off-peak times in [Chiaraviglio et al., 2009b]. They demonstrated that an energy saving of at least 23% is possible for the total energy consumed by the backbone network. Chiaraviglio et al. model a network for

minimizing the energy consumption by switching off idle nodes (routers) and idle links, witch subjects to flow conservation and maximum link utilization constraints in [Chiaraviglio et al., 2008]. The problem is NP-hard, so in [Chiaraviglio et al., 2009a], several simple heuristic algorithms are employed, which sort all the nodes depending on the number of links, the number of flows they accommodate, or use a random strategy to switch off for saving energy. In a simple network scenario, which includes core, edge, and aggregate networks, it is possible to switch off 30% of links and 50% of nodes. But most of them do not consider some practical problems of the on/on approach: switching on and off takes time, it leads to a network reconfiguration because of topology change, and a wake-up method is required to determine how and when nodes and links should be switched on again. Bianzino et al. extend their work by considering a real-world case in [Bianzino et al., 2010]. The problem for minimizing the number of nodes and links for saving energy, given a certain traffic demand, has been solved by Integer Problem Formulation (ILP) for simple networks. The algorithm switches off devices that consume power in descending order. The results show that it is possible to reduce power consumption by more than 23%, that is, 3 GWh/year in real network scenario.

The proposed framework further addresses the shortcomings of the existing approaches and proposes a sleep-based, traffic-aware power management strategy to significant reduce network power consumption through reconfiguring the network and putting the lightly loaded routers into sleep mode. In addition, sleep mode technique is considered jointly with speed scaling technique to further adjust power and energy consumption, while adhering close to QoS requirements.

5.3 Sleep-based Power Controller

In this section, we first present the basic sleep-based power controller architecture. We then discuss a sleep-based traffic-aware power management strategy, which is used to optimize the network configuration and putting into sleep mode the lightly loaded network elements, while taking into account network performance requirements.

5.3.1 Sleep-based Traffic-aware Power Controller Architecture

The proposed sleep-based power management framework assumes the existence of a *sleep-based, traffic-aware power controller* that regulates access to the network. The controller relies on the existing routing protocol infrastructure to compute routing paths between a traffic source and destination. Furthermore, the controller uses a flow's traffic specification to verify the feasibility of network reconfiguration by shutting down the router with light load along the computed routing path to accommodate the flow's QoS requirements. If successful, the new router path is reconfigured to route the traffic generated by QoS flows to its destination. Otherwise, the routing rearrangement request is rejected.

The basic idea of sleep-based traffic-aware power controllers is to dynamically put into sleep mode the lightly loaded network elements and reroute the traffic load to a new feasible nearest path, based on the current state of the network, to reduce network power consumption. To design an effective sleep-based traffic-aware power controller (STPC), several issues must be addressed. First, a strategy must be in place to determine how traffic load impacts sleep mode decisions. Second, appropriate levels of congestion granularity must be taken into consideration when putting the router into sleep mode. Traffic load thresholds and QoS requirements to model different levels of network congestion are considered in this chapter, based on which a mechanism must be in place to predict traffic load at a router along all incoming paths to decide when the device goes into sleep mode without violating QoS performance.

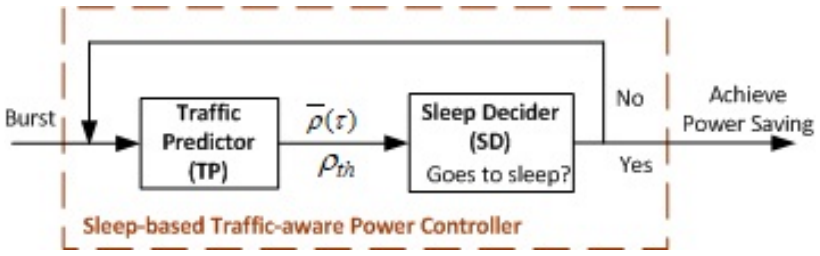

Figure 5.1: Sleep-based traffic-aware power controller architecture.

To address the above issues, a sleep-based traffic-aware power controller architecture, depicted in Figure 5.1, is proposed. The architecture has two main components: Traffic

Predictor (TP) and Sleep Decider (SD). The TP component monitors the bursts, predicts the average traffic load, $\overline{\rho}(\tau)$, over an interval τ, and gathers statics related to the state of the network. The estimated average traffic load, $\overline{\rho}(\tau)$, is used to adjust the Network Processor Unit (NPU) speed to guarantee QoS performance along the new possible routing path, and, when feasible, put the router into sleep mode.

The decision to transition a route into sleep mode is determined by a traffic load threshold, ρ_{th}. If the router predicts that its load will decrease below ρ_{th} in the next time interval, it issues a sleep request to its immediate neighbors, informing them of its intention to sleep. Depending on the feasibility of rearranging all traffic routed through the sleeping candidate, the request is either approved or denied. The details of the sleep strategy, along with supporting algorithms, are discussed next.

5.3.2 A Sleep-based, Traffic-aware Power Management Strategy

Based on the above architecture, a sleep-based traffic-aware power management strategy, referred to as STAPM, is proposed, which uses predicted average traffic load to decide whether a network device is eligible to be put into sleep mode. This strategy, including the traffic load prediction mechanism and the related sleep-based algorithms are introduced in the following.

STAPM is a strategy to shutdown under-utilized non-edge routers with the very light load to save power. It begins by predicting the traffic load based on historical records. The traffic load prediction mechanism uses the exponentially weighted moving average (EWMA) algorithm to predict the average burst bandwidth, $\overline{b}(\tau_k)$, over the k^{th} time interval τ, where $k \geq 1$. Assume $b(\tau_k)$ represents the burst occupied bandwidth over the time interval τ_k.

$$\overline{b}(\tau_k) = (1 - w_a(\tau_k)) \cdot \overline{b}(\tau_{k-1}) + w_a(\tau_k) \cdot b(\tau_k) \tag{5.1}$$

Let $\overline{b}(\tau_k)$ and $D(\tau_k)$ denote the number of predicted burst arrivals and departures over the interval τ_k, respectively. The predicted average traffic load $\overline{\rho}(\tau_k)$ can be further expressed as:

$$\overline{\rho}(\tau_k) = \frac{\overline{b}(\tau_k)}{D(\tau_k)} \tag{5.2}$$

The estimated average traffic load, $\overline{\rho}(\tau)$, is not only used to adjust the network processor unit (NPU) speed to guarantee QoS performance along the new possible routing path but also used to decide, given a traffic load threshold, ρ_{th}, whether a lightly loaded network device is eligible to move into sleep mode. If the condition, $\overline{\rho}(\tau) \leq \rho_{th}$, is satisfied for all traffic flows routed through the candidate router, the router becomes eligible to go to sleep. It then starts the attempt to transition into sleep mode by informing its immediate neighbors. In the following, we further discuss the algorithms used to migrate eligible routers into sleep mode to achieve power saving.

5.3.3 Departure Handler Algorithm

Under-utilized routers are selected as candidates to move into sleep mode to reduce power consumption for the network. If the traffic loads, passing through the candidate route, can be rerouted, the most power-efficient combination paths is selected and all traffic is rerouted to the appropriate destination. The router then moves into sleep mode. If the traffic can't be rerouted, the router's attempt to sleep fails, and the router remains in its current state. This is shown in Departure Handler Algorithm in Algorithm 5.5 in Section 5.4.1.2.

It is worth mentioning that it is a heuristics to determine the router to sleep by the traffic prediction before the current departure event is handled. The traffic prediction might appear to be higher than the actual load and is thus more likely to guarantee the low utilization rate for a period.

It should be noted that the router will be directly put into sleep mode if there is no traffic at the instant when the Departure Handler Algorithm is called. Shutting down the whole router might cause the network to be disconnected after the removal of the links adjacent to the sleep mode. The consequence is that the bursts arrived after this instant will be automatically blocked. This case is extremely likely to happen if the traffic load is low.

The following provides more discussion on the sleep request/response procedure and the related issues in detail.

Algorithm 5.1 SleepRequest().

Data: Network status NS, a candidate router r, neighbor routers, $Nb(r)$
Return: true/false
Initialization:
$SleepFlag \leftarrow$ false, $m \leftarrow 1$, $k \leftarrow ||Nb(r)||$
if r is NOT empty **then**
    r **sends sleep requests to all neighbors,** $Nb(r)$
    while $m \leq k$ **do**
        for Each $m \in Nb(r)$ **do**
            r **waits for the response from** m
            SleepResponse(NS, r**,** m**)**
            if r gets an approval from m **then**
                $m++$
            else
                r tries later
            end if
        end for
        if $m = k + 1$ **then**
            r **is approved from all neighbors**
            $SleepFlag \leftarrow$ true
            r informs all neighbors the request is approved
            r sets a wake-up time, $min\,(I_1, \cdots, I_k)$
            r goes to sleep
        else
            $SleepFlag \leftarrow$ false
            r tries later
        end if
        Return: SleepFlag
    end while
end if
$SleepFlag \leftarrow$ false

Algorithm 5.2 SleepResponse().

1: **Data:** Network status NS, the neighbor router m of r
2: **Return:** yes/No
3: **Dispatch message from sleep request queue in** m
4: m wakes up to process request from r
5: **while** Sleep request queue is NOT empty **do**
6: Process the sleep request from r
7: **if** All flow at m routed through r are rearrangeable **then**
8: **Return:** yes
9: Send a wakeup time interval, I_m, to r
10: Send a lease message
11: **else**
12: **Return:** no
13: **end if**
14: Process the next request in the queue after r's request
15: **end while**

5.3.4 Sleep Control Algorithms

The network is defined as a graph $N = (R, L)$, where R ($||R|| = n$, and $R = E_{router} \cup O_{router}$) represents the set of routers, which is composed of tow sets: the set of edge routers, E_{router} and the set of non-edge routers, O_{router}, and $L = \{(i, j)|i, j \in R \& i \neq j\}$ represents the set of links between routers. In the network $N = (R, L)$, when a non-edge router, $r \in O_{router}$, determines that it can go to sleep based on the predicted load and the related threshold, r sends sleep request to all its neighbors, $Nb(r)$. $Nb(r)$ is the set of all neighbors of the router, r, and $||Nb(r)|| = k$ ($k < n$). Then r waits for approvals from all its neighbors, meanwhile, r continues to route traffic from all of its neighbors to their destinations. If r is approved by all neighbors with 'yes', r goes to sleep, otherwise, r tries later. In other words, once r's sleep request is approved, r informs all of its neighbors that the request is approved, sets a wake-up time, and then goes to sleep. The wake-up time is greater than and equal to the sleep time to save energy, which is greater than the energy used to wake up. Every neighbor, $m \in Nb(r)$, where $1 \leq m \leq k$, sends back a sleep time interval, I_m, to predict when traffic

increases to exceed the traffic load threshold. The router sleep time is set to be equal to the minimal value of I_m $(1 \leq m \leq k)$, i.e. $min\,(I_1, \cdots, I_k)$. Sleep Request Algorithm in Algorithm 5.1 and Sleep Response Algorithm in Algorithm 5.2 further describe the sleep control algorithms from two sides: the router side with light traffic load and its neighbor side responsible for dealing with sleep requests.

Let $\mathcal{F}_r(m)$ be the set of traffic flows at neighbor router, $m \in Nb(r)$, routed though router, r, and $\mathcal{F}_m$ be set of traffic flows at m. $\mathcal{F}_r(m) = \{f \in \ \mathcal{F}_m \ (m \in Nb(r)) \,|f \ is \ routed \ through \ r\}$, where $1 \leq m \leq k$. Before approving r's sleep request, m makes sure that all $f \in \mathcal{F}_r(m)$ can be rerouted to their destinations through alternative new paths without router r in network N, while adhering to the QoS requirements.

The above description is the general sleep request/response procedure, the real situation is more complex. For example, when multiple sleep requests are sent to the same neighbor from different routers, this neighbor can not deal with multiple requests simultaneously. Thus, it puts the received sleep requests into a queue according to the time sequence and obeys with the first in first out (FIFO) policy to check the earliest one among the requests sent by these different routers at first. Assuming that two routers, r_1 and r_2, where $r_1, r_2 \in O_{router}$, determine they can go to sleep according to the current network state. r_1 and r_2 send the sleep requests to the same neighbor, m_1, at t_1 and t_2, respectively, where $t_1 < t_2$. According to the FIFO policy, it checks the earlier request from the router, r_1, firstly. If r_1's request is approved, r_1 waits to go to sleep until all other neighbors also approve its request. But if r_1's request to m_1 is denied, r_2's request is checked by m_1 in turn. In other words, no matter that r_1's request is approved or denied, m_1 gives a lease message, then moves to check r_2's sleep request based on the queued order. Besides, another possible problem is that if there exists the third router, r_3, also sends a sleep request to another same neighbor, m_2, with the router, r_1, at time t_3. If $t_1 < t_3$, check r_1's request first according to the policy, otherwise, r_3's request first, after it is responded by m_2 , then r_1's turn. After that, repeat the above described similar procedure of dealing with r_1 and r_2's sleep requests at m_1. The different neighbor routers have their individual queue to store the received sleep requests following the time order.

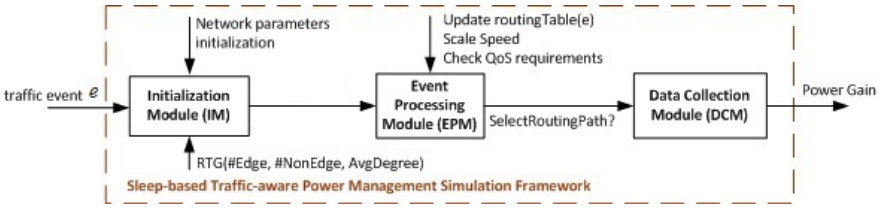

Figure 5.2: Traffic-aware power management simulation framework.

5.4 Performance Evaluation

In order to assess the performance of the proposed sleep-based traffic-aware power management strategy, STAPM, we develop a simulation framework to carry out a set of simulation-based experiments. In this framework, we consider a set of sleep-enable routers and present a detailed model to determine the *router-based* and *network-based* power consumption, taking into consideration the QoS requirements.

5.4.1 A Traffic-aware Power Management Simulation Framework

To study the performance, a traffic-aware power management simulation framework, depicted in Figure 5.2, is proposed in this section. It includes three components, namely Initialization Module(IM), Event Processing Module (EPM), and Data Collection Module (DCM). The IM component is used to set the network and traffic parameters along with other network configurations. It also initializes all the traffic events. The EPM component conducts an event by event simulation including traffic arrival, routing, bandwidth allocation and free. The DCM component processes all the data depicting the process of the sleep-based traffic-aware power management strategy, STAPM, which is generated in the EPM component. The output of DCM, namely power gain (PG), is the statistics of the processed data.

Table 5.1: Main simulation parameters and conditions.

Items	Simulation Parameters	Simulation Values
Network	Topology	$N(R, L)$
	E_{router}	A set of integers representing the edge routers
	O_{router}	A set of integers representing the non-edge routers
	R	$E_{router} \cup O_{router}$
	L	the set of links between routers
		$\{(i, j)\|i, j \in R \& i \neq j\}$
	n	$\|\|R\|\|$
	G_b	$n \times n$ bandwidth matrix
	G_d	$n \times n$ delay matrix
	P_r^e	the power cost of router r at event e
	G_b^e	the available bandwidth in $N(R, L)$ at event e
	$RoutingTable(e)$	a table records the path of the bursts currently being served all over the network at event e based on the time scale
Traffic	Burst-based flow	poisson distribution
	ρ_{th}	the lowest traffic load threshold to make router sleep

5.4.1.1 Initialization Module (IM) The Initialization Module component generates traffic events and initializes network parameters. We begin by introducing the construction process of a traffic burst along with its parameters. A traffic event is essentially a burst of traffic between a source vertex and a destination vertex (s/d pair) in the network. It can be represented with a tuple containing the following parameters (assume the current burst id is k):

- s_k: an integer representing the source vertex of burst, k,
- d_k: an integer representing the destination vertex of burst, k,
- t_a^k: a sequence number for the arrival time of burst, k,
- t_d^k: a sequence number for the departure time of burst, k,

- b_k: the bandwidth of burst, k.

The arrival time and departure time of the bursts between the same s/d pair could be generated using two distributions separately.

Assume that a series of bursts are generated from the same source/destination pair from distributions f_a^{sd} and f_b^{sd}. The bandwidth of a burst is fixed. Then there are three parameters defined at the traffic level:

- f_a^{sd}: Poisson distribution of burst arrival interval between an s/d pair.
- f_d^{sd}: Poisson distribution of burst service time between an s/d pair.

In IM component, the traffic events can be generated by sorting the t_a^k and t_d^k for all bursts. By walking through the sorted time array, mark the sequence of arrival and departure for each burst. This sequence is used in the EPM component, which is introduced in the next subsection.

The network parameters are a collection of data that describes the network. We begin by the parameters depicting the property of a network defined as a graph $N = (R, L)$, where R ($||R|| = n$, and $R = E_{router} \cup O_{router}$), represents the set of routers which includes edge routers, E_{router}, and non-edge routers, O_{router}, and $L = \{(i,j)|i,j \in R \& i \neq j\}$ represents the set of links between routers. In this chapter, a random topology generator (RTG), as displayed in Figure 5.2, is designed to generate randomly topology Matrix $RTG(\#Edge, \#NonEdge, AvgDegree)$, where $\#Edge + \#NonEdge = n$, $\#Edge$ denotes the number of edge routers, $\#NonEdge$ denotes the number of routers other than edge router and $AvgDegree$ denotes the average degree of a network graph, which is a measure of how many edges are in set compared to number of vertices in set. Accordingly, the associated parameter matrices listed in Table 5.1 are defined as follows:

- G_b: A weighted graph to represent the bandwidth. It is an $n \times n$ matrix where $G_b(i,j)$ is the link bandwidth of link (i,j). It is a symmetric graph, i.e. $G_b(i,j) = G_b(j,i)$.
- G_d: A weighted symmetric graph to represent the link delay. It is an $n \times n$ matrix where $G_d(i,j)$ is the link delay of link (i,j). The delay for the non-existent link is set to *infinity*.
- E_{router}: A set of integers representing the edge routers in the network. Only edge routers can send/receive traffic. Edge routers can't be put into sleep mode for power saving.

- O_{router}: A set of integers representing the non-edge routers in the network. They do not generate traffic while only forward traffic from edge routers. They might be put into sleep mode for power saving.

Given G_b and G_d, a delay-sensitive network can be initialized. Note that these two matrices should be integrated to actually represent the same topology. Otherwise, the result may be inconsistent.

In addition to the property parameters, a series of network status variables are also defined to record the time-dependent status of the network at every event. Several typical status parameters to record the status of the network at every event are listed as following.

- P_r^e: The power cost of the router r at event e, where $r \in R$, and $r : 1, ..., n$.
- G_b^e: The available bandwidth in the graph at event e. It is an $n \times n$ symmetric matrix in which $G_b^e(i, j)$ is the available bandwidth in link (i, j) at event e.
- $RoutingTable(e)$: This variable is a table recording the path of the bursts currently being served all over the network at event e. Each row represents a burst and has five elements: source, destination, bandwidth, burst id, and its routing path shown as a sequence of vertices. It records the routing paths at the simulation timelines at the network.

5.4.1.2 Event Processing Module (EPM) The Event Processing Module component takes the initialized network topology, traffic load, and event sequence as the input. It walks through the events according to the sequence, and process the events by handling burst arrival and burst departure. Its logic is shown in Event Sequence Handling Algorithm in Algorithm 5.3.

Upon burst arrival, the *K-shortest* path algorithm [Eppstein, 1998; Hershberger et al., 2007; Madkour et al., 2017] is conducted to find a pool with K paths. To conveniently implement traffic-aware routing, we use G_d as the topology to the algorithm such that the paths with low delays are preferred. Then we implement functions *CheckDelay* and *CheckBandwidth* to filter the result paths by testing whether there is enough bandwidth and whether the requirement of the en-to-end delay is satisfied. In this study, random selection(function *SelectPathRandom*) is implemented to pick up an available path to route

the burst.

When the routing path is decided, it is added to the routing table and available bandwidth value along the path in G_b^e at event e is decreased by the bandwidth of the burst. The traffic information and the path is then added to the routing table. If there is no path left after the filter functions, the burst will be marked blocked and no parameters are changed, as shown in Burst Arrival Handler Algorithm in Algorithm 5.4.

According to the predicted traffic load discussed in Section 5.3.2, $\overline{\rho}(\tau_k)$, whether the utilized router that the traffic flow is traveling through, as a sleep candidate, could go to sleep or not depends on whether a rerouting path to the destination could be found, without violating QoS performance along the rerouting path. If the traffic can't be rerouted, the router will not be put into sleep mode, as shown in Burst Departure Handler Algorithm in Algorithm 5.5.

5.4.1.3 Data Collection Module (DCM) The Data Collection Module component processes all the data depicting the process of the power and traffic-aware events, which is generated in the EPM component. The output of DCM is the statistics of the processed data, such as predicted link utility rate, power consumption, power gain and so on. It is capable to conduct a variety of analyses based on the status variables over the simulation time scale pattern to better investigate the performance.

Based on the above architecture, a sleep-based, traffic-aware power management strategy is presented, which uses predicted average traffic load to decide when to put the selected router into sleep mode and achieve the traffic grooming without QoS violation.

Algorithm 5.3 Event Sequence Handling Algorithm.

Data: Network status NS, Event e
Return: N/A
Initialization:
Obtain NS, e from the initialization module.
Initialize status parameters for this event e
if *e is an arrival event* **then**
    HandleArrival(NS, e)
else
    HandleDeparture(NS, e)
end if

Algorithm 5.4 Burst Arrival Handler Algorithm.

Data: Network status NS, Event e, Delay Threshold δ_0
Return: N/A
Initialization: K, Paths
From e, extract traffic information s(source), d(destination), b(bandwidth), $burstid$(burst id)
From NS, extract G_b^e(current available bandwidth), G_d^e(delay graph)
Paths←KShortestPath(s,d,G_d^e,K)
CheckDelay(Paths, δ_0)
CheckBandwidth(Paths, G_b^e)
if *Paths is NOT empty* **then**
    $path_0 \leftarrow$ SelectPathRandom(Paths)
    Decrease bandwidth in G_b^e by b along $path_0$
    Put s, d, $burstid$, b and $path_0$ to the RoutingTable
    Update other related parameters
end if
Mark this burst $burstid$

Algorithm 5.5 Burst Departure Handler Algorithm.

Data: Network status NS, Event e, Traffic Load Threshold ρ_{th}
Return: N/A
Initialization:
From e, extract $burstid$(burst id)
From NS, extract G_b^e(current available bandwidth b)
PredictTraffic(NS)
if Burst id $burstid$ is not blocked **then**
    Extract $path$ and b from Routing table from $burstid$
    Increase G_b^e values along the path by b
    Remove that entry in the Routing Table
end if
$r \leftarrow$**FindSleepableRouter(NS)**
if r is NOT empty **then**
    if For all traffic routed by r **then**
        Traffic Prediction
        if all predicted load satisfy $\overline{\rho}(\tau_k) \leq \rho_{th}$ & their paths are rearrangeable **then**
            r **sends sleep requests to all of its neighbors,** $Nb(r)$
            SleepRequest(NS, r**,** $Nb(r)$**)**
        else
            r tries later
        end if
    end if
    r **waits for all responses from the neighbors,** $Nb(r)$
    if r gets approvals from all of its neighbors **then**
        Rearrange the paths
        r goes to sleep
    else
        r tries later
    end if
end if

5.4.2 Router-based Power Model and Network-based Energy-efficient Metrics

5.4.2.1 Power measurement When discussing the power issue of a router/switch, two main aspects impacting power consumption need to be considered. The first, referred to as static power, arises from the bias and leakage current to support control plane, environment units, and load-independent data plane [Tucker et al., 2009; Vishwanath et al., 2014]. The second, referred to as dynamic power, results from the charging and discharging of the voltage saved in node capacitance of the circuit. Using Φ^S and Φ^D to denote static and dynamic power, respectively, the power Φ consumed by a router can be expressed as follows:

$$\Phi = \Phi^S + \Phi^D \tag{5.3}$$

According to [Vishwanath et al., 2014], the power consumption of an IP router is the sum of the power consumed by its three major subsystems, namely control plane, environmental units and data plane. Assume that $\Phi_{control}$ denotes the static power consumed by control plane, $\Phi_{environment}$ denotes the static power consumed by environment units, Φ^S_{data} denotes the static power consumed by the constant baseline components in data plane, and Φ^D_{data} denotes the dynamic power consumed by the traffic load dependent components in data plane. Accordingly, the above power components can be further expressed as:

$$\begin{aligned} \Phi^S &= \Phi_{control} + \Phi_{environment} + \Phi^S_{data} \\ \Phi^D &= \Phi^D_{data} \end{aligned} \tag{5.4}$$

5.4.2.2 A general power-aware model for router power consumption Joseph Chabarek et al. in [Chabarek et al., 2008] performed several experiments to measure the energy consumption of two different Cisco routers: GSR 12008 and 7507. Both of them include their base systems (Chassis plus router processor) and line cards, based on which it provides a generic model for router power consumption, as described in Eq. 5.5. In this model, the power consumption Φ of a router is determined by its configuration and current use. The vector X defines the chassis type of the device, the installed line cards and the configuration and traffic profile of the device. The function $\Phi^S(x_0)$ returns the power consumption of a

particular chassis type, which is from control plan and environment unit, N is the number of line cards that are active, $\varphi^D_{data}(x_{i0})$ is the dynamic cost with a scaling factor corresponding to the traffic utilization on the router, and $\varphi^S_{data}(x_{i1})$ gives the cost of the line card in a base configuration. The cost of traffic is dependent on the configuration of the router and the amount of traffic. This model is used to formulate the optimization problem for power-aware network design.

$$\begin{aligned}\Phi(X) =& \Phi^S(x_0) + \sum_{i=0}^{N}(\varphi^D_{data}(x_{i0}, x_{i1}) + \varphi^S_{data}(x_{i1})) \\ =& \underbrace{\Phi^S(x_0) + \sum_{i=0}^{N}\varphi^S_{data}(x_{i1})}_{\Phi^S(X)} + \underbrace{\sum_{i=0}^{N}\varphi^D_{data}(x_{i0}, x_{i1})}_{\Phi^D(X)}\end{aligned} \tag{5.5}$$

Assume that there are N active linecards in the router and that the network processing unit (NPU) of each linecard executes at speed, σ_i, where $1 \leq i \leq N$, then its dynamic power consumption can be roughly characterized as $\varphi^D(\sigma_i) = \gamma \cdot \sigma_i^3$. Define the static power as a fixed fraction of the router power consumed when NPUs' executing at maximum speeds, referred to υ (i.e. static power ratio) [Cui et al., 2014; Yu et al., 2015b]. Let $\vec{\sigma} = (\sigma_1, \cdots, \sigma_N)$. Hence, the power consumption of an active router in Eq. 5.5 can further be expressed as:

$$\begin{aligned}\Phi(X) &= \Phi(\vec{\sigma}) \\ &= \Phi^S(\vec{\sigma}^{max}) + \Phi^D(\vec{\sigma}) \\ &= \upsilon \cdot \gamma \cdot \sum_{i=0}^{N}(\sigma_i^{max})^3 + (1-\upsilon) \cdot \gamma \cdot \sum_{i=0}^{N}\sigma_i^3\end{aligned} \tag{5.6}$$

where $\upsilon \geq 0.7$ [Yu et al., 2015b].

5.4.2.3 Network-based energy-efficient metrics We use power gain (PG), defined in Eq. 5.7, as the energy-efficient metric to evaluate the performance of the proposed sleep-based traffic-aware power management strategy, STAPM, in the given network topology. Assume H and H' denotes the active number of routers in the initial configured and reconfigured network, respectively. And assume X and X' define the chassis type of the device, the installed line cards and the configuration and traffic profile of the device in each active

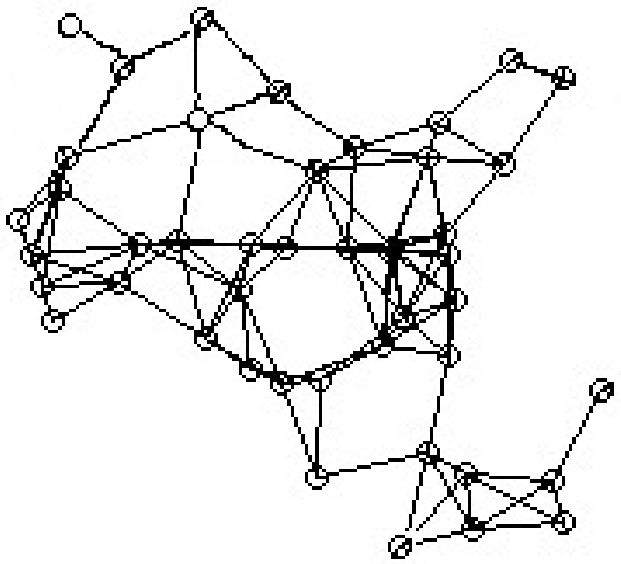

Figure 5.3: Random topology generator.

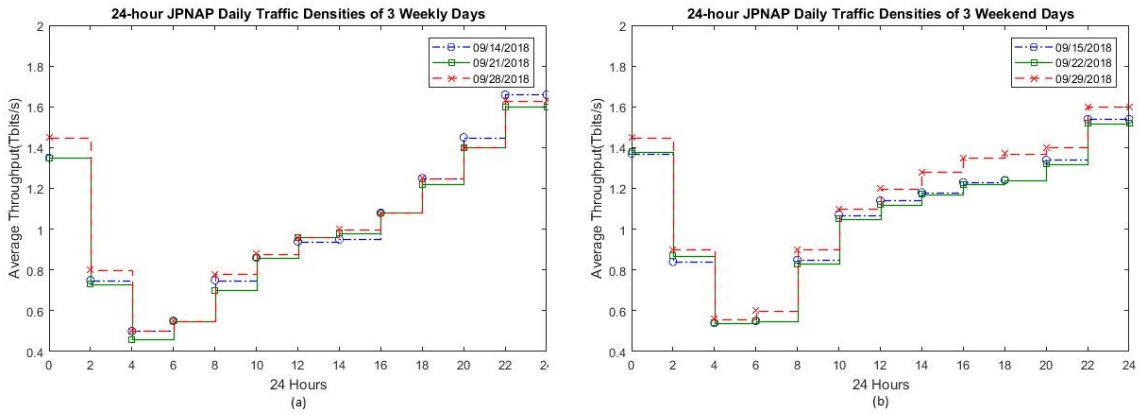

Figure 5.4: JPNAP daily traffic densities collected on (a) weekly days (Sept. 14^{th}, Sept. 21^{st}, Sept. 28^{th}, 2018); (b) weekend days (Sept. 15^{th}, Sept. 22^{th}, Sept. 29^{th}, 2018) respectively.

router in the initial configured and reconfigured network, respectively.

$$PG = \frac{\sum_{i=0}^{H} \Phi_i(X) - \sum_{i=0}^{H'} \Phi_i(X')}{\sum_{i=0}^{H} \Phi_i(X)} \tag{5.7}$$

Accordingly, another two energy-efficient metrics, namely dynamic power gain (DPG)

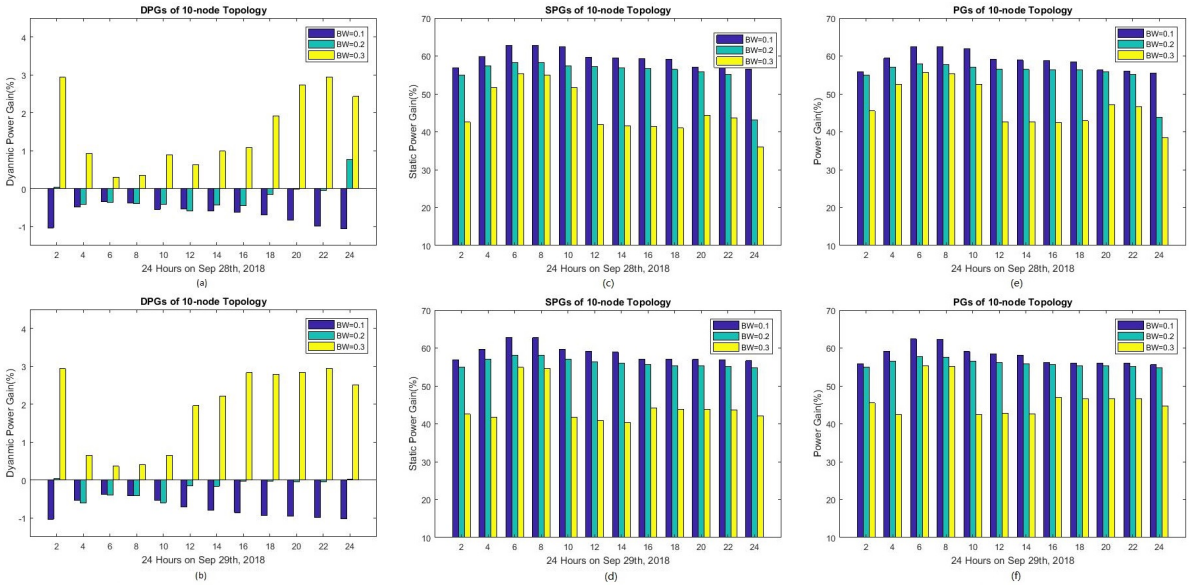

Figure 5.5: (a,b) Dynamic power gains, (c,d) Static power gains, and (e,f) Power gains of a 10-node network topology under different bandwidths on Sept. 28^{th}, 2018 (a weekly day), and Sept. 29^{th}, 2018 (a weekend day) respectively.

and static power gain (SPG) are also discussed in this chapter, they are defined as:

$$DPG = \frac{\sum_{i=0}^{H} \Phi_i^D(X) - \sum_{i=0}^{H'} \Phi_i^D(X')}{\sum_{i=0}^{H} \Phi_i^D(X)} \tag{5.8}$$

$$SPG = \frac{\sum_{i=0}^{H} \Phi_i^S(X) - \sum_{i=0}^{H'} \Phi_i^S(X')}{\sum_{i=0}^{H} \Phi_i^S(X)} \tag{5.9}$$

5.4.3 JPNAP Daily Traffic Study

In recent years, the landscape of the Internet is fast changing with the introduction of streaming content such as video, high definition television and Voice over IP (VoIP), with more stringent QoS requirements. Given the ever-increasing importance of the Internet, knowledge of its traffic has become increasingly critical. For instance, traffic properties are highly dependent on a time scale, a long time scales (hours to days to weeks) traffic exhibits strong periodicities: there are distinct diurnal and weekly cycles with traffic peaks occurring

around midday or in the evening and troughs in the early morning. This pattern correlates to user behavior, where they access the Internet during the day for work or school and sleep in the night. Thus it would be very difficult to simulate a 24-hour timeline due to the difficulty of the automatic restart process. Alternatively, we collect a series of traffic densities throughout a 24-hour from JPNAP Service [Co.], as shown in Figure 5.4. We evaluate the average power costs under each of the densities (12 samples per 24-hour) when a sleep mode is enabled. We then compare them to the power consumption at the same traffic densities when the sleep mode is disabled through the above discussed energy-efficient metrics. For either mode, the power consumption is obtained by a process simulating traffic routing throughout the network. We assume that when the traffic density changes, the network administrator will force all routing to restart and become awake.

A random topology generator is designed in this study as depicted in Figure 5.3. Each link offers fine traffic granularity which enables us to fine tune the transmission power according to the average throughput dynamically, to achieve energy-efficient burst handling. In the simulation, denote each link has bandwidth normalized to 1 and the bandwidth of a burst is typically represented by a fraction. The bandwidth usage of a link may be dynamically adjusted up and down by green rerouting, according to traffic grooming states. The dynamic power consumption of the router port connecting this link is computed by the amount of occupied bandwidth in this link. That is, the two ports from the two routers at the end of the same link should have the same dynamic power cost. Assume each link has its fixed delay and each traffic burst has a required end-to-end delay as its QoS requirement. The burst will only be routed to a routing path satisfying its delay requirement. In this chapter, three randomly generated network topologies, namely 10-node topology, 20-node topology, and 30-node topology, are discussed.

For the sleep mode simulations, it is a very hard study when a sleeping router should automatically wake up because it requires precise traffic prediction and traffic monitoring throughout the network, either of which is a complicated topic and beyond the scope of this chapter. To make the simulation process simple, we alternatively set that turned down router can not be restarted automatically, i.e. they are turned down till the end of the simulation timeline. If the traffic density increases, the network would be congested as the

sleeping routers can not wake up by themselves. To be able to handle these issues, we assume that network administrators can manually command all router back on. By setting the all-awake initial state for all routers, a fair initial status is provided to evaluate and average power consumption over the simulation timeline in different modes. By collecting and comparing the average power consumption over a long simulation timeline, we can obtain stable-state performances of our proposed power-aware mechanism at different traffic densities. What's more, this result could be applied to an arbitrary time scale and is not confined to the length of the simulation timeline. Thus we could confirm the validity of the result albeit the simulation process is not continuous over all the timelines in the experiment. Table 5.1 describes the main simulation parameters used in this simulation study. In addition, three different metrics are defined, namely *dynamic power gain* (DPG) , *static power gain* (SPG) and *power gain* (PG). The first metric is to measure the relative dynamic power gain according to the traffic grooming state after green routing. The second measures the relative static power gain due to network reconfiguration. The third metric is to measure the relative total power gain based on the former two metrics. These metrics are defined in Eq. 5.8, Eq. 5.9 and Eq. 5.7.

We further assume that the system clock in the routers of the networks are perfectly synchronized. Otherwise, the chronicle order of sleep requests in different routers may be inconsistent and cause chaos when multiple routers are sending out sleep requests in a short period of time.

5.4.4 Simulation-based Performance and Analysis

To decide whether a network router is set to sleep mode depends on how low the background traffic flow is during the given simulation timeline. Given a fixed ρ_{th}, the lowest traffic load threshold to lead a router to sleep, the objective of a set of experiments is to assess the performance of different adaptable bandwidth, such as 0.1, 0.2 and 0.3, according to traffic session changes, depicted as Figure 5.5, through the strategy $SPRAT$, based on three metrics, namely DPG, SPG, and total PG. In Figure 5.5, (a,b) shows dynamic power gains (DPGs) of a randomly generated 10-node network topology under 12 average traffic

throughput densities in 24 hours of Sept. 28^{th}, 2018 (a weekly day), and Sept. 29^{th}, 2018 (a weekend day) respectively. The traffic grooming due to green routing brings the changes of dynamic power, it might increase dynamic consumption since the traffic loads increase along the new routing paths, but the total network dynamic power might decrease due to router sleeping. Once the router goes to sleep, no dynamic power consumption at it anymore. We found DPG changes within the range of ($-1.04\% \sim 2.94\%$) are small, the reason is assuming that the green routing happens under the condition of the low traffic load in this chapter. (c,d) shows static power gains (SPGs) of this 10-node network topology, the highest SPGs are up to 62.82% and 62.77% based on the traffic densities of 6 am on Sept. 28^{th}, 2018 (a weekly day), and Sept. 29^{th}, 2018 (a weekend day) respectively. Accordingly, the highest PG is up to 62.50% and 62.39%. The more routers could be considered to be asleep from midnight to early morning, no matter whether it is a weekly day or a weekend day. Besides, we also test the blocking rates among these three bandwidth settings, the blocking rates for the bursts with fixed BW of 0.1 (normalized value as using $\frac{1}{10}$ of the link bandwidth) are less than 3% for two different days, as described in Figure 5.6, however, the other two settings lead to blocking rates exceed 20%, which are suggested not to make green routing. For all experiments are make delay filtering, once delay performance requirement is broken, the green routing is denied. Therefore, a conclusion can be drawn that the lower bandwidth setting controls lower traffic with $\rho \leq \rho_{th}$ to sleep the routers without a higher blocking rate, thereby achieving higher power saving through sleeping more routers.

Based on the above discussion, the following experiments are based on $BW = 0.1$. Figure 5.6 further shows DPGs, SPGs and PGs of the 10-node network topology on weekly days, three Fridays such as Sept. 14^{th}, Sept. 21^{st}, Sept. 28^{th}, 2018, and weekend days, three Saturdays such as Sept. 15^{th}, Sept. 22^{th}, Sept. 29^{th}, 2018 respectively. It shows the same trends among DPGs, SPGs, and PGs. We found the number of sleeping routers is higher around the time session of 4 am to 8 am, no matter whether it is a weekly day or a weekend day. The highest power savings are up to 62.58%, 62.50%, and 62.50% for three Fridays, and 62.43%, 62.43%, and 62.39% for three Saturdays, with the blocking rates less than 3%, respectively.

We further observe the power gains of bigger size network topologies such as 20-node

network topology and 30-node network topology on Sept. 28^{th}, 2018 (a weekly day), and Sept. 29^{th}, 2018 (a weekend day) respectively, as depicted in Figure 5.7. Figure 5.7(a,b) shows that the power gains of bigger size network topologies are lower than smaller size topologies because of networks' larger scale bases, and the randomly generated 20-node network topology and 30-node network topology have their power gains of up to 56.18% and 39.92% under the acceptable blocking rates, as displayed in Figure 5.7(c,d), respectively.

5.5 Conclusions

In this chapter, we proposed and developed a sleep-based, traffic-aware power management framework to save network power consumption through network reconfiguration by putting the lightly loaded network components into sleep modes, while meeting the QoS delay and the network blocking rate requirements. To this end, a simulation framework is used to assess the performance of this proposed strategy in terms of different power-efficient metrics. The simulation results show that up to 62.58% power saving of total power consumed by standard network configuration when no element is put into sleep mode can be achieved, without violating the QoS requirements. In comparison, the DVFS-based, delay-aware strategy, EDFS, discussed in Chapter 4, only achieves up to 26.76% power saving of total power consumption. While the sleep-based traffic-aware power management strategy proposed in this chapter, STAPM, has more significant power saving.

Typically, the challenge for DVFS techniques is to preserve the feasibility of the schedule and provide performance guarantees, while the challenges for the inconvenience with sleep mode techniques are that once in a power-efficient state, bringing a component back to the active or running state requires additional energy and/or delay to serve incoming traffic load, and that constantly switching network devices or their components off and on might lead to more energy consumption than keeping them on all the time. Consequently, how to more efficiently use these two DPM techniques to sufficiently reduce and even minimize network energy consumption, while adhering to QoS requirements, continues to be the goal of our further research.

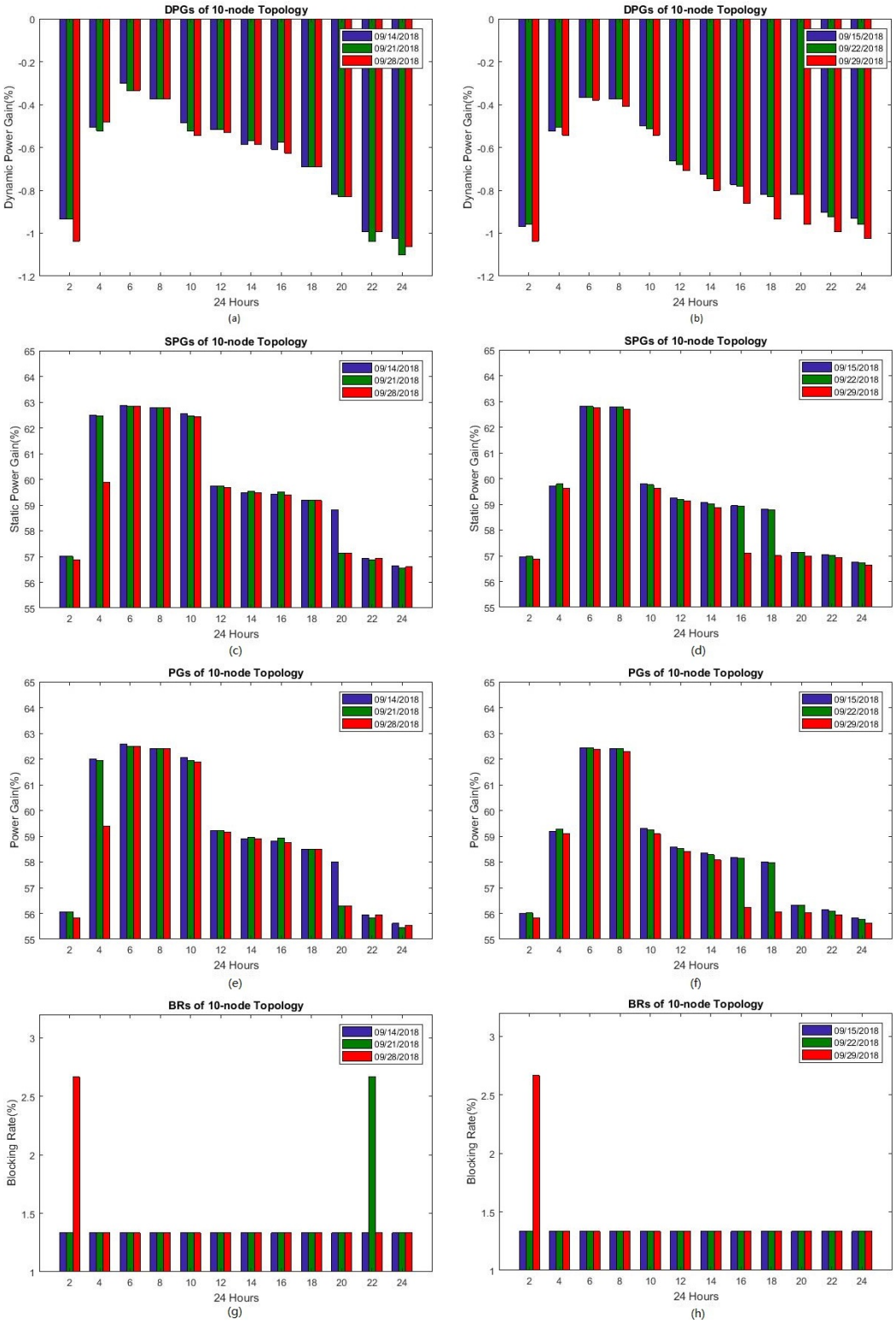

Figure 5.6: (a,b) Dynamic power gains, (c,d) Static power Gains, (e,f) Power gains, and (g,h) Blocking rates of a 10-node network topology on weekly days such as Sept. 14^{th}, Sept. 21^{st}, Sept. 28^{th}, 2018, and weekend days such as Sept. 15^{th}, Sept. 22^{th}, Sept. 29^{th}, 2018 respectively.

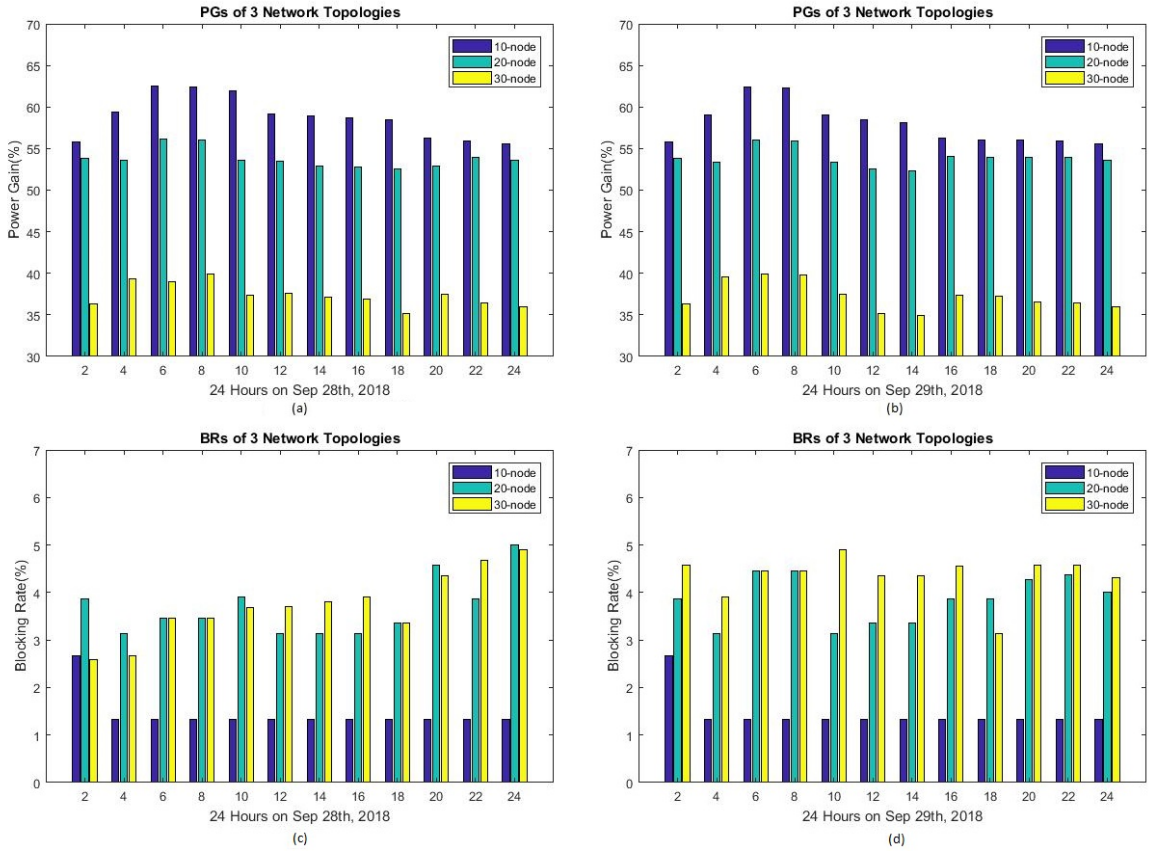

Figure 5.7: (a,b) Power gains and (c,d) Blocking rates of 10-node, 20-node, and 30-node network topologies on Sept. 28^{th},2018 (a weekly day) and Sept. 29^{th}, 2018 (a weekend day).

6.0 Conclusions and Future Work Directions

This chapter summarizes the results presented in this dissertation (Section 6.1), gives some answers to the research questions from Chapter 1 and analyzes the related limitations (Section 6.2). Finally it proposes directions for future research (Section 6.3).

6.1 Summary

In last years, the issue of energy efficiency has become of paramount importance for both the industrial and the research community, because of its potential economical benefits and expected environmental impact. Although the green networking field is still in its infancy, different research directions are already being explored. In this dissertation, we investigated and studied solutions to push energy- and QoS-awareness into wired networks, following the resource consolidation principle. During the development and implementation of the energy- and QoS-aware power management strategies, as shown in Figure 6.1, the tradeoff issue between energy saving and network performance is discussed. The topic of this dissertation is sufficient energy saving or energy minimization under QoS requirements for wired communication networks. Both experimental and algorithmic power management makes it possible for network designers and developers to significantly reduce the energy consumption of network devices. Software with various strategies can be used to decrease the speed of network elements to lower their energy consumption (speed scaling), or it can put the lightly loaded network elements in a low power sleep mode to save energy (sleep mode).

In Chapter 2, we studied the current two prominent dynamic power management techniques. Speed scaling techniques are used to modulate energy absorption according to the actual workload by slowing down network components. Sleep mode techniques can further cut the power consumption of lightly loaded devices (or parts of them) by putting these net-

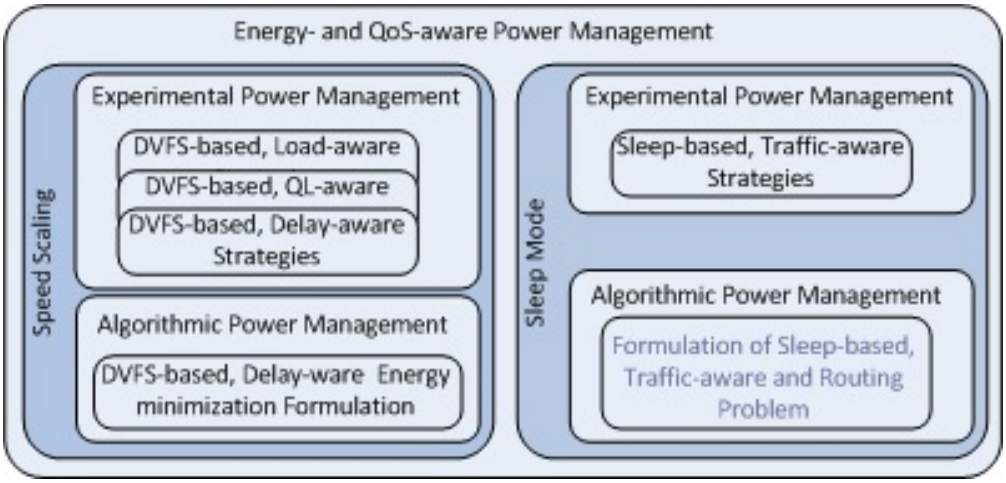

Figure 6.1: Research work.

work devices into sleep modes. These techniques may eventually impact the next-generation network devices by providing "energy-aware" profiles. As a result, seeking intelligent energy-aware strategies to make optimal decisions to switch different power-saving modes, while taking into account the tradeoff between energy saving and network performance, becomes our motivation for this research work.

In Chapter 3, we explored how to design effective QoS-aware, DVFS-based packet schedulers, based on link utilization, queue length(QL) and packet delay. And we proposed three families of DVFS-based, QoS-aware (including QL-aware and Load-aware) packet scheduling schemes [Yu et al., 2015a,b]. The best one, the QL-based, delay-aware scheme, QLDA, can achieve up to 10% dynamic energy saving of the total energy consumption in high-speed networks [Yu et al., 2015b].

In addition to DVFS-based experimental power management, we also explored dynamic power solutions based on speed scaling by algorithmic power management in Chapter 4. We studied how to formulate the energy optimization problem by using a mathematical programming model, which combines the DVFS technique with a given traffic scheduling policy. The simulation results show that minimizing energy consumption can be achieved up to 26.76% power saving of the total power consumption by using the proposed DVFS-based, delay-aware optimal energy strategy, EDFS, compared to another two heuristic methods,

under QoS constraints [Yu and Znati, 2017].

Moreover, considering the potential higher energy saving by using sleep mode approaches compared to speed scaling ones, we further explored sleep-based power management in Chapter 5. In general, many related studies have shown that the base system (including chassis, switch fabric, and router processor) consumes more than half of the whole power consumption of a network router. Therefore, switching off the whole router could save more energy than slowing down its components over the low-demand or idle periods. We focus our efforts to explore the sleep-based, energy-aware issues under QoS constraints, and proposed a sleep-based, traffic-aware power management strategy, STAPM, which focuses on how to achieve network energy maximal saving by putting lightly loaded network equipment into a low power sleep mode, thereby placing network resources in a more power- and energy-efficient way. We simultaneously considered a speed scaling technique to modulate energy absorption without damaging the QoS performance when the traffic obtain a rerouting opportunity. The results have shown that up to 62.58% (more than half) power saving of the total power consumption can be achieved by this strategy without violating the QoS requirements.

In this dissertation, our main contribution to the green networking field regards the energy- and QoS-awareness, which represented a marginally explored paradigm, while being promising in terms of achievable energy savings. Since energy savings are achieved at the price of a reduction of the redundancy level, and the offered QoS, seeking a fine tradeoff is a real challenge. To target energy-performance challenges and address the energy-QoS dichotomy, we studied different metrics, such as link utilization, queue length, packet delay, and bandwidth, and tried to strike a fine balance among performance, energy, and accuracy in Chapter 3, Chapter 4, and Chapter 5. In this dissertation, the related holistic simulation frameworks, including energy consumption models, are proposed to evaluate and compare the performance of each scheme in different networking environments and traffic models.

6.2 Conclusions

In the introduction (Chapter 1), several research questions were presented. The research in the subsequent chapters have provided answers to these questions. In the following, some main answers and limitations are summarized.

- What characteristics does a DVFS-based, QoS-aware scheduler have?

  The DVFS-Scheduler dynamically adjusts processor frequency, based on the current state of the network, to reduce energy while meeting QoS performance. To design a QoS-aware, energy-efficient strategy for speed-scaling, several issues must be addressed. The first issue is related to monitoring network traffic to determine current network congestion. This information is used to adjust processor frequency. When the congestion is high, the frequency must be scaled up to meet the QoS requirements of the application. When the network congestion is low, however, the frequency is scaled down to reduce energy consumption, without violating QoS performance. The second issue deals with accurately determining the congestion granularity and time scale needed to effectively manage frequency scaling. A finer congestion granularity measured over a short time scale leads to higher accuracy, but at the expense of additional overhead. Therefore, a tradeoff among granularity, time scale, and overhead must be worked out to achieve accuracy while maintaining a low overhead. The third issue deals with the scheduler's aggressivity when scaling the processor's frequency up or down. An aggressive strategy to lower the processor speed to save energy, when the network congestion is low, may lead to a violation of QoS performance. Similarly, an aggressive strategy to increase the processor speed in response to a high burst of traffic may lead to energy waste. The strategy must, therefore, achieve the right balance between saving energy and adhering to QoS performance.
- What characteristics does a DVFS-based, delay-aware energy minimizing schedule have?

  For DVFS-enabled network components, the key issue is how to build up their energy optimization problem, related to speed scaling, to subject to the conflicting dual objectives of speed scaling and QoS requirements. Our proposed DVFS-based algorithmic power management techniques explored this issue. To characterize the energy minimiz-

ing schedule for a given traffic flow, each router along the path must determine the node execution speed at which it processes traffic to minimize energy consumption under the QoS constraints. The energy consumption is influenced by both the schedule and the chosen speeds. In order to minimize energy consumption, both problems have to be solved simultaneously (this problem is NP-hard in most cases). In summary, to design an effective delay-aware optimal energy scheme given a scheduling policy, several issues must be addressed: (i) the development of a model to compute a path-based dynamic energy consumption; (ii) a methodology to compute feasible lower and upper bound delays of a flow, based on the router's current traffic load; and (iii) a strategy to compute feasible per-router execution rates and per-router delays that meet the end-to-end delay requirements and minimize energy consumption across the path.

- What characteristics does a sleep-based, traffic-aware power controller have? And how a sleep-based power management strategy is designed?

  To design an effective sleep-based, traffic-aware power controller, several issues must be addressed. First, a strategy must be in place to determine how traffic load impacts sleep mode decisions. Second, the routers in the network should support network granularity, such as bandwidth granularity used in the proposed strategy, in order to enable traffic estimation and power saving in sleep mode. Traffic load thresholds and QoS requirements to model different levels of network congestion are considered, based on which the designed mechanism must be in place to predict traffic load at a router along the routing path to decide when the device goes into sleep mode without violating QoS performance. The clocks in the router should be synchoronized to properly handle sleep requests in a distributed manner.

In our research, we explored and developed several energy- and QoS-aware strategies to achieve significant energy saving under QoS constraints for wired communication networks. These strategies mainly involve either reducing the transmission rate of network devices according to the load or putting unneeded network elements into sleep modes. During the development and implementation of these energy- and QoS-aware strategies, we also encountered some limitations for each scheme, which can be summarized as follows:

In experimental power management, various DVFS-based techniques are used to exploit the variations in the actual workload for dynamically adjusting the voltage and frequency of processors to achieve energy saving. In addition to the challenge for these DVFS-based techniques to preserve the feasibility of the schedule and provide performance guarantees, another challenge for these DVFS techniques is overhead issues. Energy- and QoS-awareness is imperative for green Internet design, subsequently, the low hardware HW overhead energy-awareness is becoming more and more critical. In Chapter 3, we discussed the different energy saving levels caused by different interval options, but we did not go into much depth on the overhead issue. We only analyzed two simple dumbbell and parking lot network topology models. In the real world, however, there are more realistic and complex network topologies. Therefore, given the QoS requirements, the network model, the routing path, and the character of traffic load could be important factors that impact the upper bound of some energy aggressivity parameters and could lead to more complex computing overhead issues, which should be considered in future work.

On the other hand, although algorithmic power management is a less expensive one to use mathematical models to formulate the energy minimization problem, the energy consumption, however, is influenced by both the schedule and the chosen speeds. To minimize energy consumption under QoS constraints, both problems have to be solved simultaneously (this problem is NP-hard in most cases). In Chapter 4, we only discussed the dynamic energy minimization problem, given a routing path. In the future, we will consider whether it could be a global DVFS-based energy minimization problem for a whole network.

For all DVFS-based power management techniques discussed in this dissertation, the energy saving through scaling speed is limited due to the load-dependent power source percentage occupancy among the whole power consumption. Another problem is that the energy consumption of current IP networks is not proportional to the utilization level. Therefore, even in low or no usage context, network equipment consumes energy at a high level. Compared to slowdown approaches to save energy during low-demand periods, shutdown approaches could achieve the goal of saving more energy by putting network equipment or their components into sleep mode when they are in idle states or low-demand. Thus, we provided another effective sleep-based, traffic-aware strategy to save more power consumption

without QoS violation in Chapter 5. Typically, the challenge for DVFS-based techniques is to preserve the feasibility of schedule and provide performance guarantees, while the inconvenience with sleep mode techniques is that once in a power-efficient state, bringing a component back to the active or running state requires additional energy and/or delay to serve incoming traffic load and that constantly switching network devices or their components off and on might lead to more energy consumption than keeping them on all the time. In Chapter 5, we did not delve into the disconnection issues when putting a network device into a low power sleep mode or focus on the power consumed by reacting device back from a sleep state. Our further work will involve these issues. Furthermore, whether a sleep-based, power-aware routing problem could be mathematically modeled as a global power minimization problem for a whole network will also be in our future research.

6.3 Recommendations for Future Research

With the dramatic increase in energy expenditures of the Internet, the power consumption of routers is becoming a bottleneck. Minimizing energy consumption under QoS requirements is a major concern, which has not been well-studied. The explored existing dynamic power management techniques and solutions have the potential to save significant energy, they, however, also face a drawback of the increased network latency. Speed scaling causes the stretching of packet service times because of slowing-down, while sleep mode introduces an additional delay due to wake-up times. These issues and challenges need to be overcome to establish more eco-friendly and eco-sustainable solutions. In order to improve environmental and economical sustainability, seeking intelligent strategies incorporating energy-awareness into network control and management becomes more and more critical for green Internet design. Based on the exploration of the current solutions and new research trends for future green wired networks, the conclusions indicate that research on dynamic power management for next-generation wired networks should primarily address the following aspects: (i) an effective power- or energy-aware architecture design; (ii) intelligent energy management strategies to adapt the power consumption of networks to current traffic load; (iii) accurate

energy-efficient metrics; and (iv) a fine balance in the tradeoffs between performance and energy.

Now, new algorithms for network-wide control, both distributed and centralized, are starting to take green metrics, which determine how energy efficiency will be defined, into account. For example, a possible distributed solution currently builds upon link-state protocols and puts links in an Internet protocol-based network into sleep mode at appropriate times. This method limits the amount of shared information, avoiding explicit coordination among nodes, and reducing the problem complexity. Thus, the switch-off decision considers the current workload and the history of past decisions. Furthermore, according to a green metric, energy performance thresholds, requirements or targets can be measured for defined equipment, such as energy consumption watt-hour (Wh) in relation to the delivered quality of service over defined time intervals, and energy consumption (Wh) or power requirement (W) in relation to performed traffic over defined time intervals.

Moreover, the fundamental problem of greening the Internet is to strike a fine balance between the demands of performance and the limitations of energy usage. New research initiatives in energy optimization have revealed several aspects of the Internet that can be streamlined. Addressing the issues of energy efficiency will allow us to draw deeper conclusions on how new network systems can be smarter and more effective. The solutions enable energy awareness in the Internet architecture by slowing down the speed of network elements according to the load or shutting down network elements when not in use, taking into account the tradeoff between energy saving and network performance.

Appendix

Bibliography

http://www.voip-info.org/wiki/view/Codecs.

Bernardetta Addis, Antonio Capone, Giuliana Carello, Luca G Gianoli, and Brunilde Sansò. Energy management in communication networks: a journey through modelling and optimization glasses. *Computer Communications*, 2016.

Marina Alonso, Juan Miguel Martinez, Vicente Santonja, and Pedro Lopez. Reducing power consumption in interconnection networks by dynamically adjusting link width. In *Euro-Par 2004 Parallel Processing*, pages 882–890. Springer, 2004.

Theodore P Baker. Multiprocessor edf and deadline monotonic schedulability analysis. In *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003*, pages 120–129. IEEE, 2003.

Aruna Prem Bianzino, Claude Chaudet, Federico Larroca, Dario Rossi, and Jean-Louis Rougier. Energy-aware routing: a reality check. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1422–1427. IEEE, 2010.

Aruna Prem Bianzino, Claude Chaudet, Dario Rossi, and Jean-Louis Rougier. A survey of green networking research. *Communications Surveys & Tutorials, IEEE*, 14(1):3–20, 2012.

Raffaele Bolla, Roberto Bruschi, and Andrea Ranieri. Performance and power consumption modeling for green cots software router. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1–8. IEEE, 2009.

Raffaele Bolla, Roberto Bruschi, Antonio Cianfrani, and Marco Listanti. Enabling backbone networks to sleep. *Network, IEEE*, 25(2):26–31, 2011a.

Raffaele Bolla, Roberto Bruschi, Franco Davoli, and Flavio Cucchietti. Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures. *Communications Surveys & Tutorials, IEEE*, 13(2):223–244, 2011b.

Raffaele Bolla, Franco Davoli, Roberto Bruschi, Ken Christensen, Flavio Cucchietti, and Suresh Singh. The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization? *Communications Magazine, IEEE*, 49(8):80–86, 2011c.

Raffaele Bolla, Roberto Bruschi, Alessandro Carrega, Franco Davoli, Diego Suino, Constantinos Vassilakis, and Anastasios Zafeiropoulos. Cutting the energy bills of internet service providers and telecoms through power management: An impact analysis. *Computer Networks*, 56(10):2320–2342, 2012.

Radu Carpa, Olivier Gluck, Laurent Lefevre, and Jean-Christophe Mignot. Improving the energy efficiency of software-defined backbone networks. *Photonic Network Communications*, 30(3):337–347, 2015.

Joseph Chabarek, Joel Sommers, Paul Barford, Cristian Estan, David Tsiang, and Steve Wright. Power awareness in network design and routing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1130–1138, 2008.

K Chan, R Sahita, S Hahn, and K McCloghrie. Rfc3317: Differentiated services quality of service policy information base,? *Internet Engineering Task Force*, 2003.

Xi Chen, Xue Liu, Shengquan Wang, and Xiao-Wen Chang. Tailcon: Power-minimizing tail percentile control of response time in server clusters. In *SRDS*, pages 61–70, 2012.

Luca Chiaraviglio, Marco Mellia, and Fabio Neri. Energy-aware networks: Reducing power consumption by switching off network elements. In *FEDERICA-Phosphorus tutorial and workshop (TNC2008)*, 2008.

Luca Chiaraviglio, Marco Mellia, and Fabio Neri. Reducing power consumption in backbone networks. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–6. IEEE, 2009a.

Luca Chiaraviglio, Marco Mellia, and Fabio Neri. Energy-aware backbone networks: a case study. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1–5. IEEE, 2009b.

Cisco. http://www.ici-cn.com/download/brochure/convergeIn *Converge IP and DWDM Layers in the Core Network. White paper*, 2007.

Internet Multifeed Co. Jpnap service. `http://www.jpnap.net/english/service/traffic.html`.

Xiaolong Cui, Bryan N Mills, Taieb Znati, and Rami G Melhem. Shadows on the cloud: An energy-aware, profit maximizing resilience framework for cloud computing. In *CLOSER*, pages 15–26, 2014.

Juanita Ellis, Charles Pursell, and Joy Rahman. *Voice, video, and data network convergence: architecture and design, from VoIP to wireless*. Academic Press, 2003.

David Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.

Minoru Etoh, Tomoyuki Ohya, and Yuji Nakayama. Energy consumption issues on mobile network systems. In *Applications and the Internet, 2008. SAINT 2008. International Symposium on*, pages 365–368. IEEE, 2008.

Brian Field, Taieb F Znati, and Daniel Mosse. V-net: A framework for a versatile network architecture to support real-time communication performance guarantees. In *INFOCOM'95.*, pages 1188–1196. IEEE, 1995.

Will Fisher, Martin Suchara, and Jennifer Rexford. Greening backbone networks: reducing energy consumption by shutting off cables in bundled links. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pages 29–34. ACM, 2010.

Jaime Galan-Jimenez and Alfonso Gazo-Cervero. Elee: Energy levels-energy efficiency tradeoff in wired communication networks. *Communications Letters, IEEE*, 17(1):166–168, 2013.

Marco Egbertus Theodorus Gerards. *Algorithmic power management: Energy minimisation under real-time constraints*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2014.

Ebrahim Ghazisaeedi and Changcheng Huang. Energy-efficient virtual link reconfiguration for off-peak time. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2015.

Ebrahim Ghazisaeedi, Ning Wang, and Rahim Tafazolli. Chapter5:link sleeping optimization for green virtual network infrastructures. In *2012 IEEE Globecom Workshops*, pages 842–846. IEEE, 2012.

Luca Giovanni Gianoli. Energy-aware traffic engineering for wired ip networks. 2014.

Nature Publishing Group. Nature photonics. In *Nature Photonics Technology Conference 2007*, pages 1–8. Tokyo, Japan, 2007.

Chamara Gunaratne, Ken Christensen, and Bruce Nordman. Managing energy consumption costs in desktop pcs and lan switches with proxying, split tcp connections, and scaling of link speed. *International Journal of Network Management*, 15(5):297–310, 2005.

Madhu Gupta and Sushil Singh. Dynamic ethernet link shutdown for energy conservation on ethernet links. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 6156–6161. IEEE, 2007a.

Maruti Gupta and Suresh Singh. Greening of the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26. ACM, 2003.

Maruti Gupta and Suresh Singh. Using low-power modes for energy conservation in ethernet lans. In *INFOCOM*, volume 7, pages 2451–2455, 2007b.

Maruti Gupta, Satyajit Grover, and Suresh Singh. A feasibility study for power management in lan switches. In *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pages 361–371. IEEE, 2004.

D. Hayes, D. Ros, L. Andrew, and S. Floyd. Common tcp evaluation suite, 2014. URL `https://tools.ietf.org/html/draft-irtf-iccrg-tcpeval-01`.

Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *NSDI*, volume 10, pages 249–264, 2010.

John Hershberger, Matthew Maxel, and Subhash Suri. Finding the k shortest simple paths: A new algorithm and its implementation. *ACM Transactions on Algorithms (TALG)*, 3(4):45, 2007.

Kerry Hinton, Jayant Baliga, Michael Feng, Robert Ayre, and Rodney S Tucker. Power consumption and energy efficiency in the internet. *IEEE Network*, 25(2):6–12, 2011.

Filip Idzikowski, Sebastian Orlowski, Christian Raack, Hagen Woesner, and Adam Wolisz. Saving energy in ip-over-wdm networks by switching off line cards in low-demand scenarios. In *Optical Network Design and Modeling (ONDM), 2010 14th Conference on*, pages 1–6. IEEE, 2010.

Hideaki Imaizumi and Hiroyuki Morikawa. Directions towards future green internet. In *Towards Green Ict*, volume 9, pages 37–53. River Publishers, Niels Jernes Vej 10, 9220 Aalborg, Denmark, 2010.

Intel. Data plane development kit overview, 2012. URL `http://www.intel.com/content/dam/www/public/us/en/documents/presentation/dpdk-packet-processing-ia-overview-presentation.pdf`.

ITU-T. Recommendation g. 114. *One-Way Transmission Time, Standard G*, 114, 2003.

Rodger J. Energy efficient ethernet: Technology, application and why you should care, 2014. URL `https://communities.intel.com/community/wired/blog/2011/05/05/`.

E Jonckheere, K Shah, and S Bohacek. Dynamic modeling of internet traffic for intrusion detection. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 2436–2442. IEEE, 2002.

Jonathan G Koomey et al. Estimating total power consumption by servers in the us and the world, 2007.

Sofie Lambert, Ward Van Heddeghem, Willem Vereecken, Bart Lannoo, Didier Colle, and Mario Pickavet. Worldwide electricity consumption of communication networks. *Optics express*, 20(26):B513–B524, 2012.

Christoph Lange, Dirk Kosiankowski, Christoph Gerlach, Fritz-Joachim Westphal, and Andreas Gladisch. Energy consumption of telecommunication networks. *ECOC 2009*, 2009.

Christoph Lange, Dirk Kosiankowski, Rainer Weidmann, and Andreas Gladisch. Energy consumption of telecommunication networks and related improvement options. *Selected Topics in Quantum Electronics, IEEE Journal of*, 17(2):285–295, 2011.

Sungjin Lee and Jihong Kim. Using dynamic voltage scaling for energy-efficient flash-based storage devices. In *2010 International SoC Design Conference*, pages 63–66. IEEE, 2010.

Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

C Lubritto, A Petraglia, C Vetromile, F Caterina, A D'Onofrio, M Logorelli, G Marsico, and S Curcuruto. Telecommunication power systems: energy saving, renewable sources and environmental monitoring. In

INTELEC 2008-2008 IEEE 30th International Telecommunications Energy Conference, pages 1–4. IEEE, 2008.

Amgad Madkour, Walid G Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. A survey of shortest-path algorithms. *arXiv preprint arXiv:1705.02044*, 2017.

Malcolm Mandviwalla and Nian-Feng Tzeng. Energy-efficient scheme for multiprocessor-based router linecards. In *Applications and the Internet, 2006. SAINT 2006. International Symposium on*, pages 8–pp. IEEE, 2006.

Zoltán Móczár, Stephen Molnar, and Balázs Sonkoly. Multi-platform performance evaluation of digital fountain based transport. In *Science and Information Conference (SAI), 2014*, pages 690–697. IEEE, 2014.

Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI*, volume 8, pages 323–336, 2008.

Bruce Nordman and Ken Christensen. Proxying: The next step in reducing it energy use. *Computer*, 43(1): 91–93, 2010.

Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys (CSUR)*, 46(4):47, 2014.

Michael J Panik. *Classical optimization: foundations and extensions*. North-Holland Pub. Co., 1976.

Jean-Marc Pierson. *Large-scale Distributed Systems and Energy Efficiency: A Holistic View*. John Wiley & Sons, 111 River Street, Permissions Department, Hoboken, NJ 07030, USA, 2015.

Bart Puype, Willem Vereecken, Didier Colle, Mario Pickavet, and Piet Demeester. Power reduction techniques in multilayer traffic engineering. In *Transparent Optical Networks, 2009. ICTON'09. 11th International Conference on*, pages 1–4. IEEE, 2009.

Ravishankar Rao and Sarma Vrudhula. Energy optimal speed control of devices with discrete speed sets. In *Proceedings of the 42nd annual Design Automation Conference*, pages 901–904. ACM, 2005.

Diego Reforgiato Recupero et al. Toward a green internet. *Science*, 339(6127):1533–1534, 2013.

Karthikeyan Sabhanatarajan, Ann Gordon-Ross, Mark Oden, Mukund Navada, and Alan George. Smartnics: Power proxying for reduced power consumption in network edge devices. In *Symposium on VLSI, 2008. ISVLSI'08. IEEE Computer Society Annual*, pages 75–80. IEEE, 2008.

Khushboo Shah, Stephan Bohacek, and Edmond A Jonckheere. On the predictability of data network traffic. In *Proceedings of the American Control Conference*, volume 2, pages 1619–1624, 2003.

Wes Simpson. *Video over IP: a practical guide to technology and applications*. Taylor & Francis, 2006.

Tim Szigeti and Christina Hattingh. *End-to-end qos network design*. Cisco press, 800 East 96th Street, Indianapolis, Indiana 46240, USA, 2005.

Rodney S Tucker, Rajendran Parthiban, Jayant Baliga, Kerry Hinton, Robert WA Ayre, and Wayne V Sorin. Evolution of wdm optical ip networks: A cost and energy perspective. *Journal of Lightwave Technology*, 27(3):243–252, 2009.

Giorgio Luigi Valentini, Walter Lassonde, Samee Ullah Khan, Nasro Min-Allah, Sajjad A Madani, Juan Li, Limin Zhang, Lizhe Wang, Nasir Ghani, Joanna Kolodziej, et al. An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing*, 16(1):3–15, 2013.

Nedeljko Vasić and Dejan Kostić. Energy-aware traffic engineering. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 169–178. ACM, 2010.

C Vassilakis. Towards energy efficient internet service providers econet perspective. In *GN3 Green Networking Workshop: Advances in Environmental Policy and Practice, Utrecht, Netherlands*, 2012.

Willem Vereecken, Lien Deboosere, Didier Colle, Brecht Vermeulen, Mario Pickavet, Bart Dhoedt, and Piet Demeester. Energy efficiency in telecommunication networks. In *Proceedings of NOC2008, the 13th European Conference on Networks and Optical Communications*, pages 44–51, 2008.

Arun Vishwanath, Kerry Hinton, Robert WA Ayre, and Rodney S Tucker. Modeling energy consumption in high-capacity routers and switches. *IEEE JSAC*, 32(8):1524–1532, 2014.

Arun Vishwanath Member, Kerry Hinton, Rob Ayre, and Rodney Tucker. Modeling energy consumption in high-capacity routers and switches. *Selected Areas in Communications, IEEE Journal on*, 32(8):1524–1532, 2014.

Lawrence J. Wobker. Power consumption in high-end routing systems, 2012. URL `https://www.nanog.org/meetings/nanog54/presentations/Wednesday/Wobker.pdf`.

Qun Yu and Taieb Znati. Energy and delay-aware traffic control and management in large scale networks. In *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*, pages 1–8. IEEE, 2017.

Qun Yu, Taieb Znati, and Wang Yang. Energy-efficient, qos-aware packet scheduling in high-speed networks. *Selected Areas in Communications, IEEE Journal on*, 33(12):2789–2800, 2015a.

Qun Yu, Taieb Znati, and Wang Yang. Energy-efficient, delay-aware packet scheduling in high-speed networks. In *Computing and Communications Conference (IPCCC), 2015 IEEE 34th International Performance*, pages 1–8. IEEE, 2015b.

Sherali Zeadally, Samee Ullah Khan, and Naveen Chilamkurti. Energy-efficient networking: past, present, and future. *The Journal of Supercomputing*, 62(3):1093–1118, 2012.

Baoke Zhang, Karthikeyan Sabhanatarajan, Ann Gordon-Ross, and Alan George. Real-time performance analysis of adaptive link rate. In *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, pages 282–288. IEEE, 2008a.

Guo-Qing Zhang, Guo-Qiang Zhang, Qing-Feng Yang, Su-Qi Cheng, and Tao Zhou. Evolution of the internet and its cores. *New Journal of Physics*, 10(12):123027, 2008b.

Mingui Zhang, Cheng Yi, Bin Liu, and Beichuan Zhang. Greente: Power-aware traffic engineering. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 21–30. IEEE, 2010a.

Yi Zhang, Massimo Tornatore, Pulak Chowdhury, and Biswanath Mukherjee. Time-aware energy conservation in ip-over-wdm networks. In *Photonics in Switching*, page PTuB2. Optical Society of America, 2010b.

Taieb F Znati and Rami Melhem. Node delay assignment strategies to support end-to-end delay requirements in heterogeneous networks. *Networking, IEEE/ACM Transactions on*, 12(5):879–892, 2004.